НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
Факультет физики

ОАД

«Билеты к экзамену»

Лектор: Корнилов М. В.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Москва
2021

Содержание

Замечания и благодарности

Данный конспект написан студентами и для студентов. Он может содержать опечатки, неточности и серьёзные смысловые ошибки. Над ним работали:

- Написание:

  М. Блуменау

Ссылка на записи лекций
Ссылка на репозиторий с исходными файлами

Структуры данных

Билет 1-3. О-нотация. Приведите примеры алгоритмов над структурами данных с константным и квадратичным (с линейным и логарифмическим) по числу элементов временем работы. Приведите примеры, в которых выбор асимптотически более медленного с точки зрения О-нотации алгоритма предпочтителен.

В информатике временная сложность алгоритма определяется как функция от длины строки, представляющей входные данные, равная времени работы алгоритма на данном входе. Временная сложность алгоритма обычно выражается с использованием нотации «О» большое, которая учитывает только слагаемое самого высокого порядка, а также не учитывает константные множители, то есть коэффициенты. Если сложность выражена таким способом, говорят об асимптотическом описании временной сложности, то есть при стремлении размера входа к бесконечности. Временная сложность обычно оценивается путём подсчёта числа элементарных операций, осуществляемых алгоритмом. Время исполнения одной такой операции при этом берётся константой, то есть асимптотически оценивается как $O(1)$. В таких обозначениях полное время исполнения и число элементарных операций, выполненных алгоритмом, отличаются максимум на постоянный множитель, который не учитывается в О-нотации.

Примеры алгоритмов:

- $O(1)$, константный:

  Определение чётности целого числа (представленного в двоичном виде)

- $O(n^2)$, квадратичный:

  Сортировка пузырьком (попарное сравнение соседних элементов)

  Сортировка вставками (рассматриваем по элементы по одному, каждый новый элемент - в подходящее место)

- $O(n)$, линейный:

  Поиск наименьшего или наибольшего элемента в неотсортированном массиве

- $O(log(n))$, логарифмический:

  Бинарный поиск (метод деления пополам)

Пример, когда асимптотически более медленный алгоритм предпочтителен:

Сложность вычисления произведения матриц по определению составляет $O(n^3)$, однако существуют более эффективные алгоритмы, применяющиеся для больших матриц.

Первый алгоритм быстрого умножения больших матриц был разработан Фолькером Штрассеном в 1969. В основе алгоритма лежит рекурсивное разбиение матриц на блоки $2 \cdot 2$. Штрассен доказал, что такие матрицы можно некоммутативно перемножить с помощью семи умножений, поэтому на каждом этапе рекурсии выполняется семь умножений вместо восьми. В результате асимптотическая сложность этого алгоритма составляет $O(n^{\log_2 7}) \approx O(n^{2.81})$. Недостатком данного метода является бОльшая сложность программирования по сравнению со стандартным алгоритмом, слабая численная устойчивость и бОльший объём используемой памяти. Разработан ряд алгоритмов на основе метода Штрассена, которые улучшают численную устойчивость, скорость по константе и другие его характеристики. Тем не менее, в силу простоты алгоритм Штрассена остаётся одним из практических алгоритмов умножения больших матриц.

В дальнейшем оценки скорости умножения больших матриц многократно улучшались. Однако эти алгоритмы носили теоретический, в основном приближённый характер. В силу неустойчивости алгоритмов приближённого умножения в настоящее время они не используются на практике. Лучший на данный момент - $O(n^{2.37})$.

Таким образом, можно сказать, что такие примеры в основном связаны с используемой памятью и устойчивостью.

Билет 4. Сортировка. Приведите примеры методов сортировки.

Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

- Сортировка пузырьком —- один из самых известных алгоритмов сортировки. Здесь нужно последовательно сравнивать значения соседних элементов и менять числа местами, если предыдущее оказывается больше последующего. Таким образом элементы с большими значениями оказываются в конце списка, а с меньшими остаются в начале. Этот алгоритм считается учебным и почти не применяется на практике из-за низкой эффективности: он медленно работает на тестах, в которых маленькие элементы (их называют «черепахами») стоят в конце массива. Худшее $O(n^2)$, среднее $O(n^2)$.

- Сортировка вставками – массив постепенно перебирается слева направо. При этом каждый последующий элемент размещается так, чтобы он оказался между ближайшими элементами с минимальным и максимальным значением. Худшее $O(n^2)$, среднее $O(n^2)$.

- Сортировка выбором – сначала нужно рассмотреть подмножество массива и найти в нём максимум (или минимум). Затем выбранное значение меняют местами со значением первого неотсортированного элемента. Этот шаг нужно повторять до тех пор, пока в массиве не закончатся неотсортированные подмассивы. Худшее $O(n^2)$, среднее $O(n^2)$.

- Быстрая сортировка – этот алгоритм состоит из трёх шагов. Сначала из массива нужно выбрать один элемент — его обычно называют опорным. Затем другие элементы в массиве перераспределяют так, чтобы элементы меньше опорного оказались до него, а большие или равные — после. А дальше рекурсивно применяют первые два шага к подмассивам справа и слева от опорного значения. Худшее $O(n^2)$, среднее $O(n \cdot log(n))$.

- Сортировка слиянием – пригодится для таких структур данных, в которых доступ к элементам осуществляется последовательно (например, для потоков). Здесь массив разбивается на две примерно равные части и каждая из них сортируется по отдельности. Затем два отсортированных подмассива сливаются в один. Худшее $O(n \cdot log(n))$, среднее $O(n \cdot log(n))$.

Билет 5-8. Деревья. Определения: двоичное дерево, глубина дерева, сбалансированное дерево, дерево поиска, двоичное дерево поиска, минимальное и максимальное время поиска значения. Какова глубина сбалансированного двоичного дерева? Сбалансированное двоичное дерево поиска, для чего используется, время поиска значения, пример построения из упорядоченного массива.

Дерево – одна из наиболее широко распространённых структур данных в информатике, эмулирующая древовидную структуру в виде набора связанных узлов. Является связным графом, не содержащим циклы. Большинство источников также добавляют условие на то, что рёбра графа не должны быть ориентированными. В дополнение к этим трём ограничениям, в некоторых источниках указывается, что рёбра графа не должны быть взвешенными.

Понятие графа максимально просто – имеются парные связи. Односвязный граф – любые две вершины связаны. Листьями называют вершины, из которых нельзя пойти далее. Количество переходов, которые необходимо совершить к листу называют глубиной Двоичное дерево - такое дерево, что каждый узел которого имеет не более двух потомков. Потомки – выходящие из этого узла узлы. Сбалансированное двоичное дерево - такое дерево, что для каждого узла глубина отличается не более чем на 1. Двоичное дерево поиска (англ. binary search tree, BST) – структура данных для работы с упорядоченными множествами. Двоичным деревом поиска называют двоичное сбалансированное дерево, для каждого узла которого выполняется следующее условие: $\forall\ x_i$ из правого поддерева $x_i < z$; $\forall\ x_i$ из левого поддерева $x_i > z$. Поиск займёт $O(log(n))$ (худшее $O(n)$, когда дерево не сбалансировано и представляет собой такую колбасу вытянутую), вставка и удаление – $O(log(n))$, глубина равна $log_2(n)$ Варианты сбалансированных деревьев поиска:

- АВЛ деревья
- Красно-черные деревья

АВЛ дерево: В каждом узле хочу хранить разность между высотой левого и правого поддеревьев. $\delta d = \{-1, 0, 1\}$ Есть 4 преобразования, позволяющих поправить балансировку назад и они заключаются в переписывании указателей потомков(выполняется за константное время) и их надо сделать не более чем $log(n)$ раз при попытке добавить новый лист.

- Двоичное дерево - используется в многих поисковых приложениях , где данные постоянно на входе/выходе, такие , как map и set объектах в библиотеках многих языков.
- Раздел двоичного пространства - используется почти в каждой трехмерной видеоигре, чтобы определить, какие объекты необходимо визуализировать.
- Двоичные попытки (radix tree). Используются практически в каждом маршрутизаторе с высокой пропускной способностью для хранения таблиц маршрутизатора.
- Huffman Coding Tree (Chip Uni) - используется в алгоритмах сжатия, таких как файлы форматов .jpeg и .mp3.

Базовый интерфейс двоичного дерева поиска состоит из трёх операций:

FIND(K) — поиск узла, в котором хранится пара (key, value) с key = K. INSERT(K, V) — добавление в дерево пары (key, value) = (K, V). REMOVE(K) — удаление узла, в котором хранится пара (key, value) с key = K.

Кроме того, интерфейс двоичного дерева включает ещё три дополнительных операции обхода узлов дерева: INFIX-TRAVERSE, PREFIX-TRAVERSE и POSTFIX-TRAVERSE. Первая из них позволяет обойти узлы дерева в порядке неубывания ключей.

Поиск элемента (FIND) Дано: дерево T и ключ K. Задача: проверить, есть ли узел с ключом K в дереве T, и если да, то вернуть ссылку на этот узел. Алгоритм:

Если дерево пусто, сообщить, что узел не найден, и остановиться. Иначе сравнить K со значением ключа корневого узла X. Если K=X, выдать ссылку на этот узел и остановиться. Если K>X, рекурсивно искать ключ K в правом поддереве T. Если K<X, рекурсивно искать ключ K в левом поддереве T.

Добавление элемента (INSERT) Дано: дерево T и пара (K, V). Задача: вставить пару (K, V) в дерево T (при совпадении K, заменить V). Алгоритм:

Если дерево пусто, заменить его на дерево с одним корневым узлом ((K, V), null, null) и остановиться. Иначе сравнить K с ключом корневого узла X. Если K>X, рекурсивно добавить (K, V) в правое поддерево T. Если K<X, рекурсивно добавить (K, V) в левое поддерево T. Если K=X, заменить V текущего узла новым значением.

Удаление узла Дано: дерево T с корнем n и ключом K. Задача: удалить из дерева T узел с ключом K (если такой есть). Алгоритм:

Если дерево T пусто, остановиться; Иначе сравнить K с ключом X корневого узла n. Если K>X, рекурсивно удалить K из правого поддерева T; Если K<X, рекурсивно удалить K из левого поддерева T; Если K=X, то необходимо рассмотреть три случая. Если обоих детей нет, то удаляем текущий узел и обнуляем ссылку на него у родительского узла; Если одного из детей нет, то значения полей ребёнка m ставим вместо соответствующих значений корневого узла, затирая его старые значения, и освобождаем память, занимаемую узлом m; Если оба ребёнка присутствуют, то Если левый узел m правого поддерева отсутствует (n->right->left) Копируем из правого узла в удаляемый поля K, V и ссылку на правый узел правого потомка. Иначе: Возьмём самый левый узел m, правого поддерева n->right; Скопируем данные (кроме ссылок на дочерние элементы) из m в n; Рекурсивно удалим узел m.

Обход дерева (TRAVERSE) Есть три операции обхода узлов дерева, отличающиеся порядком обхода узлов. Первая операция — INFIX-TRAVERSE — позволяет обойти все узлы дерева в порядке возрастания ключей и применить к каждому узлу заданную пользователем функцию обратного вызова f, операндом которой является адрес узла. Эта функция обычно работает только с парой (K, V), хранящейся в узле. Операция INFIX-TRAVERSE может быть реализована рекурсивным образом: сначала она запускает себя для левого поддерева, потом запускает данную функцию для корня, потом запускает себя для правого поддерева.

INFIX-TRAVERSE (tr) — обойти всё дерево, следуя порядку (левое поддерево, вершина, правое поддерево). Элементы по возрастанию PREFIX-TRAVERSE (tr) — обойти всё дерево, следуя порядку (вершина, левое поддерево, правое поддерево). Элементы, как в дереве POSTFIX-TRAVERSE (tr) — обойти всё дерево, следуя порядку (левое поддерево, правое поддерево', вершина). Элементы в обратном порядке, как в дереве.

INFIX-TRAVERSE Дано: дерево T и функция f Задача: применить f ко всем узлам дерева T в порядке возрастания ключей Алгоритм:

Если дерево пусто, остановиться. Иначе: Рекурсивно обойти левое поддерево T. Применить функцию f к корневому узлу. Рекурсивно обойти правое поддерево T. В простейшем случае функция f может выводить значение пары (K, V). При использовании операции INFIX-TRAVERSE будут выведены все пары в порядке возрастания ключей. Если же использовать PREFIX-TRAVERSE, то пары будут выведены в порядке, соответствующем описанию дерева, приведённого в начале.

Билет 9. k-мерное двоичное дерево поиска. Время поиска значения, примеры использования.

k-d-дерево (англ. k-d tree, сокращение от k-мерное дерево) — это структура данных с разбиением пространства для упорядочивания точек в k-мерном пространстве. k-d-деревья используются для некоторых приложений, таких как поиск в многомерном пространстве ключей (поиск диапазона и поиск ближайшего соседа). k-d-деревья — особый вид двоичных деревьев поиска. Поиск лучший – $O(log(n))$, средний $O(n)$.

K-мерное дерево — это несбалансированное дерево поиска для хранения точек из $\mathbb{R}^k$. Оно предлагает похожую на R-дерево возможность поиска в заданном диапазоне ключей. В ущерб простоте запросов, требования к памяти $O(kn)$. Существуют однородные и неоднородные k-d-деревья. У однородных k-d-деревьев каждый узел хранит запись. При неоднородном варианте внутренние узлы содержат только ключи, листья содержат ссылки на записи.

Очевидно, что минимальное количество просмотренных элементов равно 1, а максимальное количество просмотренных элементов — $O(h)$, где h — это высота дерева.

Средняя величина считается по формуле: $A_n = \sum_{k=1}^{n} k p_{n,k}$ Остаётся найти вероятность $p_{n,k}$. Она равна $p_{n,k} = \frac{p_{A,k}}{p_n}$, где $p_{A,k}-$ число случаев, когда $A = k$ и p_n- общее число случаев. Не сложно догадаться, что $p_{n,k} = \frac{2^{k-1}}{2^n-1}$. Подставляем это в формулу для средней величины: $A_n = \sum_{k=1}^{n} k p_{n,k} = \sum_{k=1}^{n} k\frac{2^{k-1}}{2^n-1} = \frac{1}{2^n-1}\sum_{k=1}^{n} k2^{k-1} = = \frac{1}{2^n-1}\sum_{k+1=1}^{n}(k+1)2^k = \frac{1}{2^n-1}\left(\sum_{k+1=1}^{n} k2^k + \sum_{k+1=1}^{n} 2^k\right) = = \frac{1}{2^n-1}\left(\sum_{k=1}^{n} k2^k + \sum_{k=1}^{n} 2^k - 2^n - n2^n\right) = = \frac{1}{2^n-1}\left(n2^{n+2} - (n+1)2^{n+1} + 2 - 2^n + 2^3 - 1 - n2^n\right) = \frac{2^n(n-1)+1}{2^n-1}$ то есть $A_h = \frac{2^h(h-1)+1}{2^h-1}$, где $h-$ высота дерева. Если перейти от высоты дерева к количеству элементов, то: $A_n = O\left(\frac{2^h(h-1)+1}{2^h-1}\right) = O\left(h\frac{2^h}{2^h-1} - 1\right) = O\left(\log\left(\frac{n}{N}+1\right)\frac{2^{\log\left(\frac{n}{N}+1\right)}}{2^{\log\left(\frac{n}{N}+1\right)}-1} - 1\right) = O\left(\log\left(\frac{n}{N}+1\right)\frac{n+N}{n} - 1\right) = = O\left(\log\left(\frac{n}{N}+1\right)^{\frac{n+N}{n}} - 1\right)$, где $N-$ количество элементов в узле.

Из этого можно сделать вывод, что чем больше элементов будет содержаться в узле, тем быстрее будет проходить поиск по дереву, так как высота дерева будет оставаться минимальной, однако не следует хранить огромное количество элементов в узле, так как при таком способе всё дерево может выродиться в обычный массив или список.

Логично, что такое разбиение обычно используют для сужения диапазона поиска в K-мерном пространстве. Например, поиск ближнего объекта (вершины, сферы, треугольника и т.д.) к точке, проецирование точек на 3D сетку, трассировка лучей (активно используется в Ray Tracing) и т.п. При этом объекты пространства помещаются в специальные параллелепипеды-bounding box-ы(bounding box-ом назовем такой параллелепипед, который описывает исходное множество объектов или сам объект, если мы строим bounding box лишь для него. У точек в качестве bounding box-а берется bounding box с нулевой площадью поверхности и нулевым объемом), стороны которых параллельны осям координат.

Хеш-таблицы.

Можем эффективно искать элемент вектора по его индексу.Представим, что мы решили хранить числа с плавающей точкой. Есть массив, в котором есть какая-то связь между элементом и его индексом. Например, я храню число в ячейке с номером, который соответствует самому этому числу. Но это требует большого количества памяти. Можем ввести функцию $h(x) \to i \in [0, M]$, сопоставляющую каждому элементу по значению какой-то номер.

Тогда мы можем завести массив размера M, получится за константное время находить кусочек памяти, где лежит тот или иной элемент. Такая функция называется хэш-функцией при соблюдении ряда условий:

Детерминированность (всегда есть элементы, для которых значение функции совпадает, если мы хотим вставить элемент куда-то, а там уже есть другое значение, то такая ситуация называется словом коллизия)

Если есть какое-то распределение P(x), то если мы применим к нему хорошую функцию, то должно получиться равномерное распределение. Тогда вероятность коллизий наименее возможна. Если что-то совпало, то можем начать хранить в ячейках не просто элементы, а список элементов, для которых хэш совпадает. Проблема: либо в каждой ячейке много элементов, и поиск стремится к линейному времени, или мы расширяем таблицу и у нас будет некоторое количество ячеек, где не хранится ничего(пустой список).

Пример: $h = [M\{KA\} -] -$

$A \in [0, 1]$ - действительное

K - то, что мы хотим хэшировать.

Для строк можно придумать что-то подобное (например, взять сумму)

Таким образом можно хранить и словари.

Если кратко: хотим модифицировать вектор, чтобы находить его элемент не по его номеру, а по значению.

Для этого находим функциональную связь между номером и значением. (Пример с гардеробщицей и хэш - функцией)

Главное помнить про то, что функция должна возвращать примерно равномерное распределение

В самом общем случае все форматы файлов, с которыми мы встретимся - это некоторая формализованная пара прямого и обратного отношения F, F^{-1} между тем множеством данных в множество конечных последовательностей. Обычно такое формализованное правило или алгорим задается в текстовом виде, где написано, как мы должны записывать некий файл в каком-то формате.

Всегда можно придумать свой формат, который будет читать именно ваша программа.

Когда мы говорим про какой-то формат хранения данных, то мы имеем ввиду отображение в байтовое представление, которое хранится на диске и предоставляется по требованию.

Например, есть фотоаппарат, делаем из него спектрограф, измеряющий массив 1024*1024 пикселя, значения пропорциональны световому потоку.

Примерно можно поделить форматы на текстовые - условно можно открыть в блокноте (не в ворде) и увидеть там текст - и бинарные.

Чем удобны текстовые форматы? Можем прочитать глазами и написать руками. Но они не особо эффективны с точки зрения занимаемого пространства.

Дальше возникает такая проблема: допустим, есть большая табличка с бинарными данными, допустим, пусть будут снимки поверхности солнца в разных полосах. А что, если мы ходим получить срез по определенному значению на грани "кубика"данных? Мы могли бы вычислять смещение, зная ширину и высоту. Если у нас есть какая-то табличка очень большого размера, мы хотим найти что-то конкретное и взять все данные про это. Если мы не знаем, где лежит нужное имя, то придется все прочесть... Как избежать этого?

Давайте возьмем дерево или хэш таблицу. Куда-то я запишу индекс - структура даннаых: ключ - имя чего-то, значение - смещение в целевом файле.

O(log N) или O(1) можем найти элемент и прочитать его значение, т.к. это дерево. Как только мы его прочитали, то мы можем сказать системе считать только с нужного момента наш файл, тогда полная сложность будет O(log N) или O(1). Можем хотеть искать не только по имени, но и по каким-то значениям.

Базы данных:

- MySQL

- PostgreSQL
- Oracle

Реляционная алгебра - релялционные базы данных.

Алгебра - сущности и операции над ними, в результате которых получаются те же сущности, т.о. у нас есть некие отношения - табличка определенного формата из колонок. Каждая колонка имеет имя - атрибут - и тип, типы колонок могут быть разными. Строчки там называются кортежами или значениями. Табличка не обязательно должна быть материализована на диске.

Самая простая операция это выборка или селекция - операция, которая делает из отношения отношение, в котором не хватает ненужных строк. По какому-то критерию выбираем нужные отношения.

Проекция - выкидываем ненужные столбцы данных.

Соединение - из двух отношений делаем одно.

Пусть есть табличка со звездами и их спектральным классом

Есть вторая табличка с классами и их эффективной температурой.

Хотим отобрать звезды по температуре. Воображаем, что мы дописываем колонку с эффективной температурой. Получается новое отношение с бОльшим количеством колонок. Можем сделать выборку или селекцию и т.д.

Внутри сервера баз данных существует алгоритм построения других алгоритмов - планировщик, смотря на запрос и индексы строит алгоритм, какой индекс читать и по какой колонке выбирать. Надо сформулировать на понятном языке то, что хотим получить. А то, как делать, разбирается сервер баз данных.

В качестве индексов используются либо б-деревья (у каждого узла примерно 1000 потомков - с диска удобнее читать кусочки данных не меньше какого-то числа) или ЛСМ-деревья.

Git - пример хэш-таблицы. Ключ - хэш файла.

Статистика

Билет 1-3. Формулировка задачи линейной регрессии. Идея и постановка задачи определения параметров линии регрессии методом наименьших квадратов. Идея и постановка задачи определения параметров линии регрессии методом максимального правдоподобия. Сравните метод максимального правдоподобия и метод наименьших квадратов. В каких случаях они применяются, как выбрать тот или иной метод.

Линейная регрессия (англ. Linear regression) — используемая в статистике регрессионная модель зависимости одной (объясняемой, зависимой) переменной y от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) x с линейной функцией зависимости.

Модель линейной регрессии является часто используемой и наиболее изученной в эконометрике. А именно изучены свойства оценок параметров, получаемых различными методами при предположениях о вероятностных характеристиках факторов, и случайных ошибок модели. Предельные (асимптотические) свойства оценок нелинейных моделей также выводятся исходя из аппроксимации последних линейными моделями. Необходимо отметить, что с эконометрической точки зрения более важное значение имеет линейность по параметрам, чем линейность по факторам модели.

Метод наименьших квадратов (МНК,англ. LS - Least Squares) — математический метод, применяемый для решения различных задач, основанный на минимизации суммы квадратов отклонений некоторых функций от искомых переменных. Он может использоваться для «решения» переопределенных систем уравнений (когда количество уравнений превышает количество неизвестных), для поиска решения в случае обычных (не переопределенных) нелинейных систем уравнений, для аппроксимации точечных значений некоторой функции. МНК является одним из базовых методов регрессионного анализа для оценки неизвестных параметров регрессионных моделей по выборочным данным.

Пусть регрессионная зависимость является линейной: $y_t = \sum_{j=1}^{k} b_j x_{tj} + \varepsilon = x_t^T b + \varepsilon_t$ наблюдении, по столбцам - вектор значений данного фактора во всех наблюдениях). Матричное представление линейной модели имеет вид: $y = Xb + \varepsilon$ Тогда вектор оценок объясняемой переменной и вектор остатков регрессии будут равны $\hat{y} = Xb, \quad e = y - \hat{y} = y - Xb$ соответственно сумма квадратов остатков регрессии будет равна $RSS = e^T e = (y - Xb)^T (y - Xb)$ Дифференцируя эту функцию по вектору параметров b и приравняв производные к нулю, получим систему уравнений (в матричной форме): $\left(X^T X\right) b = X^T y$ расшифрованной матричной форме эта система уравнений выглядит следующим образом

$$\begin{pmatrix} \sum x_{t1}^2 & \sum x_{t1}x_{t2} & \sum x_{t1}x_{t3} & \dots & \sum x_{t1}x_{tk} \\ \sum x_{t2}x_{t1} & \sum x_{t2}^2 & \sum x_{t2}x_{t3} & \dots & \sum x_{t2}x_{tk} \\ \sum x_{t3}x_{t1} & \sum x_{t3}x_{t2} & \sum x_{t3}^2 & \dots & \sum x_{t3}x_{tk} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{tk}x_{t1} & \sum x_{tk}x_{t2} & \sum x_{tk}x_{t3} & \dots & \sum x_{tk}^2 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_k \end{pmatrix} = \begin{pmatrix} \sum x_{t1}y_t \\ \sum x_{t2}y_t \\ \sum x_{t3}y_t \\ \vdots \\ \sum x_{tk}y_t \end{pmatrix},$$

где все суммы берутся по всем допустимым значениям Если в модель включена константа (как обычно), то $x_{t1} = 1$ при всех t, поэтому в левом верхнем углу матрицы системы уравнений находится количество наблюдений n, а в остальных элементах первой строки и первого столбца - просто суммы значений переменных: $\sum x_{tj}$ и первый элемент правой части системы $-\sum y_i$ Решение этой системы уравнений и дает общую формулу МНК-оценок для линейной модели:

$$\hat{b}_{OLS} = \left(X^T X\right)^{-1} X^T y = \left(\frac{1}{n} X^T X\right)^{-1} \frac{1}{n} X^T y = V_x^{-1} C_{xy}$$

Для аналитических целей оказывается полезным последнее представление этой формулы (в системе уравнений при делении на n, вместо сумм фигурируют средние арифметические). Если в регрессионной модели данные центрированы, то в этом представлении первая матрица имеет смысл выборочной ковариационной матрицы факторов, а вторая — вектор ковариаций факторов с зависимой переменной. Если кроме того данные ещё и нормированы на СКО (то есть в конечном итоге стандартизированы), то первая матрица имеет смысл выборочной корреляционной матрицы факторов, второй вектор — вектора выборочных корреляций факторов с зависимой переменной.

Немаловажное свойство МНК-оценок для моделей с константой — линия построенной регрессии проходит через центр тяжести выборочных данных, то есть выполняется равенство:

$$\bar{y} = \hat{b_1} + \sum_{j=2}^{k} \hat{b}_j \bar{x}_j$$

В частности, в крайнем случае, когда единственным регрессором является константа, получаем, что МНК-оценка единственного параметра (собственно константы) равна среднему значению объясняемой переменной. То есть среднее арифметическое, известное своими хорошими свойствами из законов больших чисел, также является МНК-оценкой — удовлетворяет критерию

минимума суммы квадратов отклонений от неё. без матричной алгебры). Система уравнений имеет вид:

$$\begin{pmatrix} n & \sum_{t=1}^{n} x_t \\ \sum_{t=1}^{n} x_t & \sum_{t=1}^{n} x_t^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{t=1}^{n} y_t \\ \sum_{t=1}^{n} x_t y_t \end{pmatrix}$$

Отсюда несложно найти оценки коэффициентов:

$$\begin{cases} \hat{b} = \frac{n\sum_{t=1}^{n} x_t y_t - \left(\sum_{t=1}^{n} x_t\right)\left(\sum_{t=1}^{n} y_t\right)}{n\sum_{t=1}^{n} x_t^2 - \left(\sum_{t=1}^{n} x_t\right)^2} \\ \hat{a} = \frac{\sum_{t=1}^{n} y_t - \hat{b}\sum_{t=1}^{n} x_t}{n}. \end{cases}$$

случае речь идёт о модели $y = bx$. В этом случае вместо системы уравнений имеем единственное уравнение

$$\left(\sum x_t^2\right) b = \sum x_t y_t$$

Следовательно, формула оценки единственного коэффициента имеет вид

$$\hat{b} = \frac{\sum_{t=1}^{n} x_t y_t}{\sum_{t=1}^{n} x_t^2}$$

Идея метода максимального правдоподобия заключается в следующем. Составляется функция правдоподобия, которая равна совместной плотности вероятности наблюдений зависимой переменной при фиксированных значениях (реализации) независимой переменной $\mathrm{p}\left(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_\mathrm{n}/\mathrm{x}_1, \mathrm{x}_2,\right.$ Эта функция зависит от теоретических значений параметров α, β, σ^2 модели, которые неизвестны. Однако можно определить их оценки, максимизируя функцию правдоподобия по этим параметрам. Тем самым мы определим параметры плотности, при которых вероятность получения выборки наблюдений $\mathbf{y}_1$, (отсюда и название - функция правдоподобия). При выполнении условий Гаусса - Маркова совместная функция плотности вероятностей $\mathrm{p}\left(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_\mathrm{n}/\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{n}, \alpha, \beta, \sigma^2\right)$ равна произведению плотностей отдельных наблюдений $\mathrm{p}\left(\mathrm{y}_\mathrm{t}/\mathrm{x}_\mathrm{t}, \alpha, \beta, \sigma^2\right)$ (это верно, поскольку случайные переменные u, согласно третьему условию Гаусса-Маркова, независимы в разных наблюдениях). Далее, поскольку случайная переменная u подчиняется нормальному распределению, то и $\mathrm{p}\left(\mathrm{y}_\mathrm{t}/\mathrm{x}_\mathrm{t}, \alpha, \beta, \sigma^2\right)$ будет плотностью нормального распределения. С учетом этого, функцию правдоподобия можно представить в виде

$$\mathrm{L}\left(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n, \alpha, \beta, \sigma^2\right) = \mathrm{p}\left(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_\mathrm{n}/\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{n}, \alpha, \beta, \sigma^2\right) =$$

$$= (2\pi)^{-n/2}\left(\sigma^2\right)^{-n/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{t-1}^{n}\left(y_t - \alpha - \beta x_t\right)^2\right\}$$

Так как логарифм функции правдоподобия InL достигает максимума при тех же значениях искомых параметров, что и сама функция L, то удобнее максимизировать функцию

$$\ln L\left(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n, \alpha, \beta, \sigma^2\right) = -\tfrac{\mathrm{n}}{2}\ln(2\pi) - \tfrac{\mathrm{n}}{2}\ln\left(\sigma^2\right) - \\ -\tfrac{1}{2\sigma^2}\sum_{\mathrm{t}-1}^{\mathrm{n}}\left(\mathrm{y}_\mathrm{t} - \alpha - \beta\mathrm{x}_\mathrm{t}\right)^2$$

Необходимые условия максимума функции правдоподобия InL имеют вид

$$\begin{gathered} \tfrac{\partial \ln \mathrm{L}}{\partial \alpha} = \tfrac{1}{\sigma^2}\sum_{t=1}^{\mathrm{n}}\left(\mathrm{y}_\mathrm{t} - \alpha - \beta\mathrm{x}_\mathrm{t}\right) = 0 \\ \tfrac{\partial \ln \mathrm{L}}{\partial \beta} = \tfrac{1}{\sigma^2}\sum_{t=1}^{\mathrm{n}}\mathrm{x}_\mathrm{t}\left(\mathrm{y}_\mathrm{t} - \alpha - \beta\mathrm{x}_\mathrm{t}\right) = 0 \\ \tfrac{\partial \ln \mathrm{L}}{\partial \sigma^2} = -\tfrac{\mathrm{n}}{2\sigma^2} + \tfrac{1}{2\sigma^4}\sum_{t=1}^{\mathrm{n}}\left(\mathrm{y}_\mathrm{t} - \alpha - \beta\mathrm{x}_\mathrm{t}\right) = 0 \end{gathered}$$

Их решение дает следующие оценки максимального правдоподобия коэффициентов α и β

$$\widehat{\alpha} = \bar{y} - b\bar{x}, \widehat{\beta} = \frac{\sum_{i=1}^{n}(x_t - \bar{x})(y_t - \bar{y})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}$$

которые совпадают с оценками наименьших квадратов α и β. Получаем оценку максимального правдоподобия дисперсии σ^2 :

$$\widehat{\sigma}^2 = \frac{1}{\mathrm{n}}\sum_{\mathrm{t}=1}^{\mathrm{n}} \mathrm{e}_\mathrm{t}^2$$

Эта оценка отличается от несмещенной оценки наименьших квадратов $\mathrm{s}^2 = \sum_{\mathrm{t}=1}^{\mathrm{n}} \mathrm{e}_\mathrm{t}^2/(\mathrm{n}-2)$ и, следовательно, является смещенной. При больших n обе оценки дают близкий результат и при неограниченном возрастании n сходятся к истинному значению дисперсии, то есть оценка максимума правдоподобия, так же как оценка наименьших квадратов, является состоятельной.

Билет 4-5. Метод максимального правдоподобия: покажите как методом максимального правдоподобия получить параметр пуассоновского распределения по набору измерений.

Выберем фиксированное число $\lambda > 0$ и определим дискретное распределение, задаваемое следующей функцией вероятности:

$$p(k) \equiv \mathbb{P}(Y = k) = \frac{\lambda^k}{k!}e^{-\lambda}$$

где - $k-$ количество событий,

- λ - математическое ожидание случайной величины (среднее количество событий за фиксированный промежуток времени),

- $k!$ обозначает факториал числа k,

- $e = 2,718281828\ldots$ основание натурального логарифма.

Тот факт, что случайная величина Y имеет распределение Пуассона с математическим ожиданием λ, записывается: $Y \sim \mathrm{P}(\lambda)$

Пусть есть выборка $X_1, \ldots, X_n$ из распределения $\mathbb{P}_\theta$, где $\theta \in \Theta$ - неизвестные параметры. Пусть $L(\mathbf{x} \mid \theta) : \Theta \to \mathbb{R}-$ функция правдоподобия, где $\mathbf{x} \in \mathbb{R}^n$. Точечная оценка

$$\hat{\theta}_{\text{МП}} = \hat{\theta}_{\text{МП}}(X_1, \ldots, X_n) = \underset{\theta\in\Theta}{\operatorname{argmax}} L(X_1, \ldots, X_n \mid \theta)$$

правдоподобия при фиксированной реализации выборки. области определения, максимум любой функции $L(\theta)$ является максимумом функции $\ln L(\theta)$, и наоборот. Таким образом,

$$\hat{\theta}_{\text{МП}} = \underset{\theta\in\Theta}{\operatorname{argmax}} l(X_1, \ldots, X_n \mid \theta)$$

Если функция правдоподобия дифференцируема, то необходимое условие экстремума - равенство нулю её градиента:

$$g(\theta) = \frac{\partial l(\mathbf{x}, \theta_0)}{\partial\theta} = 0$$

Достаточное условие экстремума может быть сформулировано как отрицательная определённость гессиана - матрицы вторых производных:

$$H = \frac{\partial^2 l(\mathbf{x}, \theta_0)}{\partial\theta\partial\theta^T}$$

Важное значение для оценки свойств оценок метода максимального правдоподобия играет так называемая информационная матрица, равная по определению:

$$I(\theta) = E\left[g(\theta)g(\theta)^T\right]$$

оптимальной точке информационная матрица совпадает с математическим ожиданием гессиана, взятым со знаком минус:

$$I = -E\left(H_0\right)$$

Функция правдоподобия в математической статистике — это совместное распределение выборки из параметрического распределения, рассматриваемое как функция параметра. При этом используется совместная функция плотности (в случае выборки из непрерывного распределения) либо совместная вероятность (в случае выборки из дискретного распределения), вычисленные для данных выборочных значений. Если распределение вероятности зависит от параметра θ , то, с одной стороны, можно рассматривать условную вероятность событий x при заданном параметре θ , а с другой стороны — вероятность заданного события X при различных значениях параметра θ . Первый случай соответствует функции, зависящей от события

$P(x) = P(x \mid \theta)$, а второй — функции, зависящей от параметра

θ при фиксированном событии X: $L(\theta) = L(\theta \mid x = X)$. Последнее выражение и есть функция правдоподобия и показывает, насколько правдоподобно выбранное значение параметра θ при известном событии X.

Неформально: если вероятность позволяет предсказывать неизвестные результаты, основанные на известных параметрах, то правдоподобие позволяет нам оценивать неизвестные параметры, основанные на известных результатах. Поэтому для практических вычислений предпочитают использовать логарифм функции правдоподобия. - Функция $L(\mathbf{x} \mid \theta)$, где $L(\mathbf{x} \mid \theta) = \ln f_{\mathbf{X}}(\mathbf{x} \mid \theta)$ называется логарифмйческой функцией правдоподоббия - Если выборка независима, то $f_{\mathbf{X}}(\mathbf{x} \mid \theta) = \prod_{i=1}^{n} f_X\left(x_i \mid \theta\right)$ где $f_X(\cdot \mid \theta)-$ плотность или функция вероятности распределения $\mathbb{P}_\theta$. Логарифмическая функция правдоподобия в этом случае имеет вид: $L(\mathbf{x} \mid \theta) = \sum_{i=1}^{n} \ln f_X\left(x_i \mid \theta\right)$

Билет 6-9. Проверка статистических гипотез. Нулевая и альтернативная гипотеза. Ошибки первого и второго рода. Идея t-критерия Стьюдента. Идея χ^2-критерия Пирсона. Идея критерия Колмогорова.

Статистическая гипотеза (statistical hypothesys) — это определённое предположение о распределении вероятностей, лежащем в основе наблюдаемой выборки данных. Проверка статистической гипотезы (testing statistical hypotheses) — это процесс принятия решения о том, противоречит ли рассматриваемая статистическая гипотеза наблюдаемой выборке данных. Статистический тест или статистический критерий — строгое математическое правило, по которому принимается или отвергается статистическая гипотеза. Пусть задана случайная выборка $x^m = (x_1, \ldots, x_m)$ — последовательность m объектов из множества X. Предполагается, что на множестве X существует некоторая неизвестная вероятностная мера $\mathbb{P}$. Методика состоит в следующем. Формулируется нулевая гипотеза H_0 о распределении вероятностей на множестве X. Гипотеза формулируется исходя из требований прикладной задачи. Чаще всего рассматриваются две гипотезы — основная или нулевая H_0 и альтернативная H_1. Иногда альтернатива не формулируется в явном виде; тогда предполагается, что H_1 означает «не H_0». Иногда рассматривается сразу несколько альтернатив. В математической статистике хорошо изучено несколько десятков «наиболее часто встречающихся» типов гипотез, и известны ещё сотни

специальных вариантов и разновидностей. Примеры приводятся ниже. Задаётся некоторая статистика (функция выборки) $T : X^m \to \mathbb{R}$, для которой в условиях справедливости гипотезы H_0 выводится функция распределения F(T) и/или плотность распределения p(T). Вопрос о том, какую статистику надо взять для проверки той или иной гипотезы, часто не имеет однозначного ответа. Есть целый ряд требований, которым должна удовлетворять «хорошая» статистика T. Вывод функции распределения F(T) при заданных H_0 и T является строгой математической задачей, которая решается методами теории вероятностей; в справочниках приводятся готовые формулы для F(T); в статистических пакетах имеются готовые вычислительные процедуры. Фиксируется уровень значимости — допустимая для данной задачи вероятность ошибки первого рода, то есть того, что гипотеза на самом деле верна, но будет отвергнута процедурой проверки. Это должно быть достаточно малое число $\alpha \in [0, 1]$. На практике часто полагают $\alpha = 0.05$. На множестве допустимых значений статистики T выделяется критическое множество Ω_αнаименее вероятных значений статистики T, такое, что $\mathbb{P}\{T \in \Omega_\alpha \,|H_0\} = \alpha$. Вычисление границ критического множества как функции от уровня значимости α является строгой математической задачей, которая в большинстве практических случаев имеет готовое простое решение. Собственно статистический тест (статистический критерий) заключается в проверке условия: если $T(X^m) \in \Omega_\alpha$, то делается вывод «данные противоречат нулевой гипотезе при уровне значимости α». Гипотеза отвергается. если $T(X^m) \notin \Omega_\alpha$, то делается вывод «данные не противоречат нулевой гипотезе при уровне значимости α». Гипотеза принимается. Итак, статистический критерий определяется статистикой T и критическим множеством Ω_α, которое зависит от уровня значимости α. Замечание. Если данные не противоречат нулевой гипотезе, это ещё не значит, что гипотеза верна. Тому есть две причины. По мере увеличения длины выборки нулевая гипотеза может сначала приниматься, но потом выявятся более тонкие несоответствия данных гипотезе, и она будет отвергнута. То есть многое зависит от объёма данных; если данных не хватает, можно принять даже самую неправдоподобную гипотезу. Выбранная статистика T может отражать не всю информацию, содержащуюся в гипотезе H_0. В таком случае увеличивается вероятность ошибки второго рода — нулевая гипотеза может быть принята, хотя на самом деле она не верна. Допустим, например, что H_0 = «распределение нормально»; $T(X^m)$ = «коэффициент асимметрии»; тогда выборка с любым симметричным распределением будет признана нормальной. Чтобы избегать таких ошибок, следует пользоваться более мощными критериями. Широкое распространение методики фиксированного уровня значимости было вызвано сложностью вычисления многих статистических критериев в докомпьютерную эпоху. Чаще всего использовались таблицы, в которых для некоторых априорных уровней значимости были выписаны критические значения. В настоящее время результаты проверки гипотез чаще представляют с помощью достигаемого уровня значимости. Достигаемый уровень значимости (пи-величина, англ. p-value) — это наименьшая величина уровня значимости, при которой нулевая гипотеза отвергается для данного значения статистики критерия T: $p(T) = \min\{\alpha : T \in \Omega_\alpha\}$, где Ω_α — критическая область критерия. Другая интерпретация: достигаемый уровень значимости p(T) — это вероятность при справедливости нулевой гипотезы получить значение статистики, такое же или ещё более экстремальное, чем T. Если достигаемый уровень значимости достаточно мал (близок к нулю), то нулевая гипотеза отвергается. В частности, его можно сравнивать с фиксированным уровнем значимости; тогда альтернативная методика будет эквивалентна классической. Ошибка первого рода или «ложная тревога» (англ. type I error, α error, false positive) — когда нулевая гипотеза отвергается, хотя на самом деле она верна. Вероятность ошибки первого рода: $\alpha = \mathbb{P}\{T \in \Omega_\alpha | H_0\}$. Ошибка второго рода или «пропуск цели» (англ. type II error, β error, false negative) — когда нулевая гипотеза принимается, хотя на самом деле она не верна. Вероятность ошибки второго рода: $\beta(H_1) = \mathbb{P}\{T \notin \Omega_\alpha | H_1\}$.

Мощность критерия: $1-\beta(H) = \mathbb{P}\{T \in \Omega_\alpha | H\}$ — вероятность отклонить гипотезу H_0, если на

самом деле верна альтернативная гипотеза H. Мощность критерия является числовой функцией от альтернативной гипотезы H. Несмещённый критерий: $1-\beta(H) \geq \alpha$ для всех альтернатив H или, что то же самое, $\mathbb{P}\{T \in \Omega_\alpha | H\} \geq \mathbb{P}\{T \in \Omega_\alpha | H_0\}$для всех альтернатив H. Состоятельный критерий: $\beta(H) \to 0$ при $m \to \infty$для всех альтернатив H. Равномерно более мощный критерий. Говорят, что критерий с мощностью $1-\beta(H)$ является равномерно более мощным, чем критерий с мощностью$1-\beta'(H)$, если выполняются два условия: $\beta(H_0) = \beta'(H_0)$; $\beta(H_1) \leq \beta'(H_1)$ для всех рассматриваемых альтернатив $H_1 \neq H_0$, причём хотя бы для одной альтернативы неравенство строгое.

t-критерий Стьюдента — общее название для статистических тестов, в которых статистика критерия имеет распределение Стьюдента. Наиболее часто t-критерии применяются для проверки равенства средних значений в двух выборках. Нулевая гипотеза предполагает, что средние равны (отрицание этого предположения называют гипотезой сдвига). Все разновидности критерия Стьюдента являются параметрическими и основаны на дополнительном предположении о нормальности выборки данных. Поэтому перед применением критерия Стьюдента рекомендуется выполнить проверку нормальности. Если гипотеза нормальности отвергается, можно проверить другие распределения, если и они не подходят, то следует воспользоваться непараметрическими статистическими тестами. Сравнение выборочного среднего с заданным значением Задана выборка $x^m = (x_1, \ldots, x_m),\ x_i \in \mathbb{R}$. Дополнительное предположение: выборка простая и нормальная. Нулевая гипотеза $H_0:\ \bar{x} = \mu$(выборочное среднее равно заданному числу μ). Статистика критерия: $t = \frac{(\bar{x}-\mu)\sqrt{m}}{s}$ имеет распределение Стьюдента с m-1 степенями свободы, где $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$— выборочное среднее, $s^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2$— выборочная дисперсия.

Применяется для проверки нулевой гипотезы:

$H_0 : E(X) = m$ о равенстве математического ожидания E(X) некоторому известному значению m. Очевидно, при выполнении нулевой гипотезы $E(\overline{X}) = m$. С учётом предполагаемой независимости наблюдений $V(\overline{X}) = \sigma^2/n$. Используя несмещённую оценку дисперсии

$s_X^2 = \sum_{t=1}^{n}(X_t - \overline{X})^2/(n-1)$получаем следующую t-статистику:

$$t = \frac{\overline{X} - m}{s_X/\sqrt{n}}.$$

При нулевой гипотезе распределение этой статистики t(n-1). Следовательно, при превышении значения статистики по абсолютной величине критического значения данного распределения (при заданном уровне значимости) нулевая гипотеза отвергается. Критерий Колмогорова-Смирнова используется для проверки гипотезы H_0: "случайная величина X имеет распределение F(x)". Классический критерий Колмогорова (иногда говорят Колмогорова-Смирнова) предназначен для проверки простых гипотез о принадлежности анализируемой выборки некоторому полностью известному закону распределения. Пусть X_n - выборка независимых одинаково распределённых случайных величин, $F_n(x)$ - эмпирическая функция распределения, F(x) - некоторая "истинная"функция распределения с известными параметрами. Статистика критерия определяется выражением: $D_n = \sup_x |F_n(x) - F(x)|$. Обозначим через H_0 гипотезу о том, что выборка подчиняется распределению $F(x) \in \mathrm{C}^1(\mathbb{X})$ Тогда по теореме Колмогорова при справедливости проверяемой гипотезы: $\forall t > 0:\quad \lim_{n\to\infty} P(\sqrt{n}D_n \leq t) = K(t) = \sum_{j=-\infty}^{+\infty}(-1)^j \mathrm{e}^{-2j^2t^2}$. Гипотеза H_0 отвергается, если статистика $\sqrt{n}D_n$ превышает квантиль распределения K_α заданного уровня значимости α, и принимается в противном случае.

Примечание: В критерии Колмогорова целесообразно использовать статистику с поправкой Большева: $\sqrt{n}D_n + 1/(6\sqrt{n})$. Распределение этой статистики при справедливости проверяемой гипотезы быстро сходится к распределению Колмогорова и при n>25 зависимостью от объема выборки можно пренебречь.

Критерийχ^2 - статистический критерий для проверки гипотезы H_0, что наблюдаемая случайная величина подчиняется некому теоретическому закону распределения. Пусть дана случайная величина X . Гипотеза H_0 : с. в. X подчиняется закону распределения F(x). Для проверки гипотезы рассмотрим выборку, состоящую из n независимых наблюдений над с.в. X: $X^n = (x_1, \cdots x_n)\,,\ x_i \in [a, b]\,,\ \forall i = 1 \ldots n$. По выборке построим эмпирическое распределение$F^*(x)$ с.в X. Сравнение эмпирического $F^*(x)$ и теоретического распределения F(x) (предполагаемого в гипотезе) производится с помощью специально подобранной функции — критерия согласия. Рассмотрим критерий согласия Пирсона (критерийχ^2): Гипотеза H_0^* : Xn порождается функцией$F^*(x)$. Разделим [a,b] на k непересекающихся интервалов $(a_i, b_i],\ i = 1 \ldots k$; Пусть n_j - количество наблюдений в j-м интервале: $n_j = \sum_{i=1}^{n} [a_j < x_i \leq b_j]$; $p_j = F(b_j) - F(a_j)$ - вероятность попадания наблюдения в j-ый интервал при выполнении гипотезы H_0^*; $E_j = np_j$- ожидаемое число попаданий в j-ый интервал; Статистика: $\chi^2 = \sum_{j=1}^{k} \frac{(n_j - E_j)^2}{E_j} \sim \chi^2_{k-1}$- Распределение хи-квадрат с k-1 степенью свободы. В зависимости от значения критерия χ^2, гипотеза H_0может приниматься, либо отвергаться: $\chi^2_1 < \chi^2 < \chi^2_2$, гипотеза H_0выполняется. $\chi^2 \leq \chi^2_1$ (попадает в левый "хвост"распределения). Следовательно, теоретические и практические значения очень близки. Если, к примеру, происходит проверка генератора случайных чисел, который сгенерировал n чисел из отрезка [0,1] и гипотеза H_0: выборка X^n распределена равномерно на [0,1], тогда генератор нельзя называть случайным (гипотеза случайности не выполняется), т.к. выборка распределена слишком равномерно, но гипотеза H_0 выполняется. $\chi^2 \geq \chi^2_2$ (попадает в правый "хвост"распределения) гипотеза H_0 отвергается.

Билет 10. Стохастические процессы. Как автокорреляционная функция позволяет их изучать.

Случаийный процесс (случаийная функция) – семейство случайных величин, индексированных некоторым параметром, чаще всего играющим роль времени или пространства. Определение случайного процесса. Пусть задано вероятностное пространство (Ω, F, P). Параметризованное семейство X случйных величин, где T – произвольное множество, называется случайной функцией. Случайный процесс описывается статистическими характеристиками, называемыми моментами.

Корреляционные функции – важнейшие характеристики случайных процессов. Используя одномерную плотность распределения вероятности W(x, t) можно получить параметры случайного процесса, которые являются усреднением по множеству (ансамблю) реализаций данного случайного процесса в каком либо его «сечении», то есть в фиксированный момент времени t. Но задание одномерной плотности распределения вероятности не дает возможность определить характер изменения случайного процесса во времени и не характеризует взаимосвязь случайного процесса в различные моменты времени. Для этого вводят понятие двумерной плотности распределения вероятности W(x1, x2, t1, t2), описывающей связь двух значений $\xi(t1)$ и $\xi(t2)$ в произвольные моменты времени t1 и t2: $W(x1, x2, t1, t2)dx1dx2 = Px1 \leq \xi(t1) \leq x1 + dx1 \cap x2 \leq \xi(t2) \leq x2 +$ С помощью двумерной плотности распределения вероятности можно определить автокорреляционную (ковариационную) функцию $B(t1, t2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x1x2W(x1, x2, t1, t2)dx1dx2$, а также автокорреляционную функцию центрированного случайного процесса $R(t, t) = \int_{-\infty}^{\infty}(x1 - \zeta(t1))(x2 - \zeta(t2))W(x1, x2, t1, t2)dx1dx2$ или $R(t1, t2) = B(t1, t2) - \zeta(t1)\zeta(t2)$. В теории статистики автокорреляционная функция сигнала (АКФ) – это степень связи сигнала S(t) с его копией, сдвинутой на величину t, иными словами: B(τ)= $\int_{-\infty}^{\infty} S(t)S(t - \tau)dt$, при $\tau = 0$: $B(0) = \int_{-\infty}^{\infty} S2(t)dt = E$. Максимальное значение автокорреляционной функции (при $\tau = 0$) равно энергии сигнала, так как сигнал полностью коррелирован сам с собой. В теории случайных

функций АКФ является корреляционным моментом двух значений одной случайной функции: $B(t1,t2) = M[X(t1) - x(t1)] * [X(t2) - x(t2)]$ Здесь x(t) = MX(t), а MX(t) – математическое ожидание. График автокорреляционной функции можно получить, отложив по оси ординат коэффициент корреляции двух функций (базовой и функции сдвинутой на величину τ) а по оси абсцисс величину τ. Если исходная функция строго периодическая, то на графике автокорреляционной функции тоже будет строго периодическая функция. Таким образом, из этого графика можно судить о периодичности базовой функции, а, следовательно, и о её частотных характеристиках. Это применяется для анализа сложных колебаний, например электроэнцефалограммы человека. Основные свойства АКФ: 1) $R(\tau) = R(-\tau)$; 2) $\sigma 2 = R(0) \geq R(\tau)$; 3) $\lim R(\tau, \tau \to \infty)=0$. Формально можно вычислить автокорреляционную функцию и для детерминированного процесса. Например, для периодической функции F(t)=a sin(ωt) автокорреляционная функция описывается следующим выражением a2 R(τ)= 2 cos($\omega\tau$). Для периодической функции, представимой рядом Фурье f (t). Таким образом, автокорреляционная функция периодической функции текущего времени t является также периодической функцией от аргумента τ – величины временного сдвига.

Случайный процесс называется стационарным, если все многомерные законы распределения зависят только от взаимного расположения моментов времени t1, t2, ..., tn, но не от самих значений этих величин. В противном случае, он называется нестационарным. значений этого же ряда. Авторегрессионный процесс порядка p(AR(p) -процесс) определяется следующим образом

$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t$$

где $a_1, \ldots, a_p$– параметры модели (коэффициенты авторегрессии), c– постоянная (часто для упрощения предполагается равной нулю), а ε_t– белый шум. Простейшим примером является авторегрессионный процесс первого порядка AR(1)-процесс:

$$X_t = c + rX_{t-1} + \varepsilon_t$$

Для данного процесса коэффициент авторегрессии совпадает с коэффициентом автокорреляции первого порядка. Другой простой процесс - процесс Юла - AR(2)-процесс: $X_t = c + a_1 X_{t-1} + a_2 X_{t-2} + \varepsilon_t$ Можно показать, что автоковариационная и автокорреляционная функции AR(p)-процесса удовлетворяют рекуррентным соотношениям: $\gamma(k) = \sum_{j=1}^{p} a_j \gamma(k-j) \quad r(k) = \sum_{j=1}^{p} a_j r(k-j)$ усложняется. Можно показать, что дисперсия ряда $\gamma(0)$ и вектор автоковариаций γ выражаются через параметры следующим образом: $\gamma(0) = \left(1 + a^T \left(C - aa^T\right)^{-1} a\right) \sigma_\varepsilon^2, \gamma = \sigma_\varepsilon^2 \left(C - aa^T\right)^{-1} a$ приравнивается к нулю. В частности, для AR(1)-процесса матрица C равна просто единице, следовательно, $\gamma(0) = \left(1 + \frac{a^2}{1-a^2}\right) \sigma_\varepsilon^2$, что соответствует вышеуказанной формуле. можно получить следующее выражение для дисперсии данного процесса: $\gamma(0) = \frac{(1-a_2)\sigma_\varepsilon^2}{(1+a_2)\left((1-a_2)^2 - a_1^2\right)}$ На практике формулы для дисперсии процесса, выраженной через параметры модели обычно не применяются, а используется следующее выражение через ковариации: $\gamma(0) = \sigma_\varepsilon^2 + \sum_{k=1}^{p} a_k \gamma(k)$ Автокорреляционная функция авторегрессионого процесса экспоненциально затухает с возможной осцилляцией (осцилляции зависят от наличия комплексных корней у выборочной частной автокорреляционной функции временного ряда. Для AR(1)-процесса автокорреляционная функция - экспоненциально затухающая функция (без осцилляций), если выполнено условие стационарности. Частная автокорреляционная функция первого порядка равна r, а для более высоких порядков равна o. Данный метод основан на идее последовательного сглаживания членов ряда полиномами, построенными для отдельных частей ряда,

и состоит в следующем. данного отрезка ряда, и так далее. Таким образом, при переходе к очередному шагу происходит как бы скольжение "окном"шириной N по всем значениям ряда Значение N удобнее выбирать нечетным, поскольку в этом случае середина скользящего интервала (точка, в которой вычисляется значение полинома) всегда окон, дает сглаженные оценки значений анализируемого ряда Итак, пусть окно содержит нечетное число членов ряда N = 2m + 1. Для удобства и без ограничения общности будем нумеровать их целыми числами оценки коэффициентов полинома, мы можем получить следующую систему уравнений:

$$\frac{\partial}{\partial \mathbf{a}_\mathrm{i}} \sum_{t=-\mathrm{m}}^{\mathrm{m}} \left(\mathrm{y_t} - \mathrm{a_0} - \mathrm{a_1 t} - \ldots - \mathrm{a_p t^p}\right)^2 = 0, \quad \mathrm{j} = 0, 1, \ldots, \mathrm{p}$$

16) Система (5.16) в нормальной форме имеет вид

$$\sum_{t=-m}^{m} t^j y_t - a_0 \sum_{t=-m}^{m} t^j - a_1 \sum_{t=-m}^{m} t^{j+1} - \ldots - a_p \sum_{t=-m}^{m} t^{j+p} = 0$$

Решая (5.17) относительно коэффициента $\mathbf{a_0}$, получим равенство вида

$$\mathrm{a_0} = \mathrm{c_0} + \mathrm{c_1 y_{-m}} + \mathrm{c_2 y_{-(m-1)}} + \ldots + \mathrm{c_{2\ m+1} y_m}$$

18) y_t. Сглаженное значение ряда в точке $t = 0$ очевидно, равно $\widehat{y_0} = f(0) = a_0$. Следовательно, сглаженное значение ряда в точке равно взвешенному среднему (с весами $\mathbf{c_j}$) его наблюдаемых значений в 2m+1 точках. Таким образом, для оценки тренда методом скользящего среднего, необходимо определить постоянные $\mathrm{c_{\ j}}$, которые зависят только от выбора т и р, и затем вычислить $\mathbf{a_0}$ по формуле (5.18)

Оптимизация

Пусть у нас есть случайный вектор $\underline{x} \in \mathbb{R}^n$, $\underline{q} \in \mathbb{R}^n$ - фиксированный вектор, сразу говорим, что его среднее равно 0: $E\underline{x} = 0$, представим что у этого вектора есть ковариационная матрица - невыроженная, строго положительно определенная, какая-то хорошая ковариационная матрица $\Sigma_{\underline{x}}$.

1) Давайте спроецируем этот многомерный вектор на какой-то вектор в этом же пространстве, проекция будет определяться скалярным произведением:

$(\underline{\mathrm{q}}, \underline{\mathrm{x}})$

Нам интересно чему будет равняться среднее и дисперсия этого скалярного произведения: $E(\underline{q}, \underline{x}) = 0$, т.к. скалярное проихведение линейно по x.

2) Дисперсия тогда выражается просто как:

$E((\underline{q}, \underline{x}) - 0)^2 = Ex_i q_i x_j q_j = q_i E x_i x_j q_j = (^1) = (\underline{q}, \Sigma \underline{q}) \to max$

Хотим максимимзироать дисперсию, потому что это соответствует максимуму "информационного содержания": $-\int p\, ln(p)\, dx = \frac{1}{2} ln 2\pi e \sigma^2$

Ограничим искомый вектор по норме: $\|q\|^2 = 1$

Будем искать с помощью множителей Лагранжа и условного экстремума:

$\frac{\partial}{\partial \underline{q}}((\underline{q}, \Sigma \underline{q}) + \lambda(1 - \|q\|^2)) = 2\Sigma \underline{q} - 2\lambda \underline{q} = 0$

$\frac{\partial}{\partial \underline{q}}(\underline{q}, \Sigma \underline{q}) = \frac{\partial}{\partial q_n} q_i, \Sigma_{ij} q_j = \delta_{in} \Sigma_{ij} q_j + q_i \Sigma_{ij} \delta_{jn}$ [2]$= 2\Sigma \underline{q}$

[1] $Ex_i x_j = \Sigma_{ij}$

[2] $\Sigma^T = \Sigma$

Второе слагаемое расписывается аналогично, подставляя вместо матрицы ковариации единичную.

Тогда получаем следующее уравнение:

$\Sigma \underline{q} = \lambda \underline{q}$

Тогда получается, что искомые вектора $\underline{q}$ это собственные вектора Σ. Собственные значения этой матрицы имеют смысл дисперсии.

Искомая функция - максимум из собственных значений матрицы ковариации.

Ответом является собственный вектор, отвечающий максимальному собственному значению.

Пусть λ отсортированы и пронумированы: $\lambda_1 > \lambda_2 > \lambda_3 > ....$

Берем $\underline{q_1}$, λ_1: тогда этому будет отвечать проекция с наибольшей дисперсией, а вектор назвается первой (главной) компонентой.

Как найти остальные:

Находим вектор $\underline{q}$, у которого дисперсия максимальна, норма единична, но при этом он ортогонален какому-то линейному подпространству: $Q^T \underline{q} = 0$. Далее решаем ту же задачу, но в ортогональном подпространстве.

Что изменится в этом случае?

Пусть $Q \in \mathbb{R}^{n \times m}$

После первого шага матрица Q состоит из одного вектора $\underline{q_1}$, после двух щагов из векторов $\underline{q_1 q_2}$ и т.д.

Тогда в нашем уравнении на поиск экстремуму добавляется еще m дополнительных множителей Лагранжа (вектор μ):

$\frac{\partial}{\partial \underline{q}}((\underline{q}, \Sigma \underline{q}) + \lambda(1 - \|q\|^2)) + (\underline{\mu}, Q^T \underline{q}) = 2\Sigma \underline{q} - 2\lambda \underline{q} + Q\underline{\mu} = 0$

$\mu = -2Q^T \Sigma \underline{q} = -2(\Sigma Q)^T \underline{q} = -2(Q\lambda)^T \underline{q} = -2\lambda Q^T \underline{q} = 0$

Таким образом получаем, что Q состоит из собственных векторов Σ, и имеется снова уравнение:

$\Sigma \underline{q} = \lambda \underline{q}$ - но в нем $\underline{\text{q}}$ - уже не все собственнные вектора, т.к. мы находимся в ортогональном подпространстве.

Все собственные вектора Σ называются направлениями главных компонент, а соответствующие им собственные значения определяют максимум возможной дисперсии вдоль этих компонент.

По мере возрастания линейного подпространства, определяемого каким-то набором главных компонент, растет информационное содержание проекции вектора x внутрь линейного подпространства. Причем оно растет среди всех линейных подпространств.

Пусть у нас также есть случайный вектор $\underline{x} \in \mathbb{R}^n$, с нулевым средним и матрицей ковариации. Спроецируем его на какое-то линейное подпространство:

$\|(\underline{q}(\underline{q}, \underline{x}) - \underline{x})\|^2$ - то, на сколько мы при этом наврали, т.е. нам меет значение то, на что мы хотим проецировать:

$E\|(\underline{q}(\underline{q}, \underline{x}) - \underline{x})\|^2 = E((q_i q_j - \delta_{ij}) x_i (q_i q_k - \delta_{ik}) x_j) = (\underline{q}, \Sigma \underline{q}) - (\underline{q}, \Sigma \underline{q}) - (\underline{q}, \Sigma \underline{q}) + Tr\Sigma = Tr\Sigma - (\underline{q}, \Sigma \underline{q}) \to min$[3]

Условию миимальности будет удовлетворять следующее уравнение:

$\Sigma \underline{q} = -\lambda' \ \lambda = -\lambda' \ \underline{q}$

Тогда нам надо минимизировать разность $\sum_j \lambda_j - \lambda_k$, зная, что собствеенные значения матрицы ковариации положительные.

Очевидно, что для этого нужно взять в качестве λ_k максимальное собствееное значение матрицы. Тогда мы получаем, что при проецировании на главные компоненты ошибка будет

[3] при $\|q\|^2 = 1$

получться минимальной.

Среди всех возможных подпространств единичной размерности это пространство обеспечивает нам наименьшую среднюю ошибку в смысле нормы квадрата разности.

Если мы теперь рассмотрим соответственно выражение

$E\|(QQ^T\underline{x} - \underline{x})\|^2 = Tr\Sigma - TrQ^T\Sigma Q$, то получится все также

Теперь мы уже ищем минимум по матрице Q. теперь условием минимальности будет $Q^TQ = I$.

Вспоминаем, что $Q \in \mathbb{R}^{n\times m}$, понимаем, что Q - это минимальное подпространство натянутое на первые m собственных векторов главных компонент, отвечающих за максимизацию информационного содержания.

QQ^T называют автоинкодером - он кодирует вектор x в то же пространство.

Пусть теперь у нас есть теперь набор из N реализаций случайного вектора. Тогда $\underline{x_i}$, где i = 1,2, ...,N, - набор чисел. Теперь мы можем записать оценку на матрицу ковариации:

$\widetilde{\Sigma} = \frac{1}{N}\sum_{i=0}^{N} x_i \otimes {}^4x_i = \frac{1}{N}X_{ij}X_{ki}{}^5 = \frac{1}{N}XX^T$, где X - матрица измерений (матрица данных)

При этом X состоит из столбцов, каждый из которых - отдельное измерение.

Хотим найти главные компоненты и Σ

Один из способов:

Любую матрицу можно представить с помощью сингулярного разложения в виде X = USV, где $UU^T = 1$ и $VV^T = 1$, $V \in \mathbb{R}^{N\times N}$, $U \in \mathbb{R}^{n\times n}$, $S \in \mathbb{R}^{n\times N}$

Тогда получаем, что:

$XX^T = US^2U^T$, при этом дисперсия $\sigma_i^2 = \lambda_i$, а U состоит из главных компонент.

$A\underline{\theta} = \underline{x}$

$\underline{\theta}$ - параметры

A - матрица параметров, задающая условия эксперимента

- наблюдения

Примеры:

- $at_i + b = x_i$
- $lnF_i = -kM_i + ln\ F_0$ - зависимость яркости звезды от параметров

Таким образом мы получим:

$\underline{\hat{x}} = \underline{x} + \underline{\varepsilon}$, при этом $E\underline{\varepsilon} = 0$

Пусть $cov\underline{\varepsilon} = \sigma^2 I$ - все ошибки одинаковые и симметричные

$\underline{\hat{\theta}} = \underset{\underline{\theta}}{argmin}\|A\underline{\theta} - \hat{\theta}\|^2$

$\hat{\theta} = (A^TA)^{-1}A^T\underline{\hat{x}}$

При этом имеем:

$E\underline{\hat{\theta}} = (A^TA)^{-1}A^T\underline{x} = \underline{\theta}$ - несмещенная оценка

$\underline{\hat{\theta}} - E\underline{\hat{\theta}} = (A^TA)^{-1}A^T\varepsilon$

$cov_{\underline{\hat{\theta}}} = ((A^TA)^{-1}A^T cov_{\underline{\varepsilon}} A(A^TA)^{-1}) = \sigma^2(A^TA)^{-1}$

$\hat{\varepsilon} = \hat{x} - A\hat{\theta}$

ε - настоящая ошибка

$\hat{\varepsilon}$ - ошибка по мнению модели

N и M - исходные размеры матрицы

$cov_{\hat{\varepsilon}} = \sigma^2(I - A(A^TA)^{-1}A^T)$

$s^2 = \frac{1}{N-M}\|\hat{\varepsilon}\|^2 = \frac{1}{N-M}(\hat{x}, (I - A(A^TA)^{-1}A^T)\hat{x})$

[4] внешнее произведение

[5] первый индекс нумерует коспоненты, а второй сам вектор

$Es^2 = \frac{1}{N-M} Tr(I - A(A^TA)^{-1}A^T)\sigma^2 I = \sigma^2 \frac{N-M}{N-M} = \sigma^2$

Если мы знаем, что измерения некореллированы, то уже не так просто все получается.

Есть ряд принципов, позволяющих оценивать неизвестные параметры наших моделей, используя какие-то измерения. Это принцип минимума невязки (МНК, например): параметры должны быть такими, что разница между предсказанным и измеренным будет наименьшей. Или же метод максимального правдоподобия: параметры должны быть такими, чтобы те наши измерения, которые мы видим, максимизировали вероятность получения таких результатов.

Допустим у меня есть функция f(x)

И есть такая точка x^*, для которой выполняется неравенство: $f(x^*) \leq f(x) \forall x \in X$, то x^* - **точка глобального минимума**.

Допустим, у меня есть x^* такая, что $f(x^*) \leq f(x)$ выполняется не для всех x, а для таких, для которых верно: $\|x - x_0^*\| \leq \varepsilon,\ \exists \varepsilon > 0$, то есть в некоторой окрестности. Такая точка является точкой локального минимума.

Важно, что если возмем g(x) = -f(x), то экстремумы у этой и исходной функции совпадут, но там, где у одной минимум, у другой будет максимум, и наоборот.

Если мы нашли алгоритм для поиска минимума, то можем тривиально модифицировать его для поиска максимума.

Итак, мы договорились, что сформулировали задачу следующим образом: дана какая-то функция в каком-то виде и мы хотим найти ее минимумы или максимумы.

Сразу есть 2 случая: $x \in \mathbb{R}^n$, то говорят о задачу безусловной оптимизации, т.е. нет никаких условий, которые ограничивают множество допустимых значений x

Если x в каком-то подмножестве, то задача условной оптимизации, и это подмножество $x \in X \in R^n$, и X определяется какими-то равенствами или неравенствами, например ограничениями на параметры.

Алгоритмы, решающие задачу оптимизации, не всегда совпадают с алгоритмами условной оптимизации.

Эффективнее брать задачи для частных случаев для частных случаев.

Выпуклое множество в аффинном или векторном пространстве — множество, в котором все точки отрезка, образуемого любыми двумя точками данного множества, также принадлежат данному множеству.

Если x_1 и $x_2 \in X$, то и $\lambda x_1 + (1-\lambda)x_2 \in X$, при $\lambda \in [0, 1]$

Выпуклая функция — это вещественнозначная функция, определенная на интервале со свойством, что ее надграфик (множество точек на графике функции или над ним) является выпуклым множеством.

Выпуклая функция определена на выпуклом множетсве.

$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$

Примеры: $e^x, x^2, |x|^2$

Обобщение - неравенство Йенсена.

Если x* — точка локального минимума (максимума) выпуклой (вогнутой) функции f(x) на выпуклом множестве X, то x* — точка глобального экстремума функции f(x) на X, т.е. f(x*) — наименьшее (наибольшее) значение f(x) на X.

Доказательство. Предположим, что функция f(x) выпукла и x* — точка локального минимума функции f(x) на X. Допустим, что x* не является точкой глобального минимума функции f(x) на X, т.е. существует точка A ∈ X такая, что $f(A) < f(x^*)$. Любая точкаx отрезка, соединяющего x* и A, представима в виде $x = (1-\lambda)x^* + \lambda A$, $\lambda \in [0, 1]$ и

$$f(x) = f((a-\lambda)x^* + \lambda A) \leq (1-\lambda) f(x^*) + \lambda f(A) = f(x^*) + \lambda \underbrace{(f(A) - f(x^*))}_{<0} \leq f(x^*)$$

так как $\lambda > 0$. Заметим, что точка x может быть сколь угодно близкой к x*, что противоречит тому, что x* — точка локального минимума функции f(x).

Необходимые и достаточные условия локального минимума:

- Если f(x*) - локальный минимум, то $\nabla f(x^*) = 0$

  При этом если $\nabla f(x) \neq 0$, то это точно не локальный экстремум.

- Если f(x*) - локальный минимум, то $(Hh, h) \geq 0 \, \forall \underline{h} \in \mathbb{R}^n$

  Или же можно сказать, что собственные значения Гессиана($H = \frac{\partial^2 f}{\partial x_i \partial x_j}$) неотрицательны.

- Если $\nabla f(x^*) = 0$ и $(Hh, h) > 0$ (собственные значения Гессиана($H = \frac{\partial^2 f}{\partial x_i \partial x_j}$) положительные), то x* - локальный минимум.

1) По количеству требуемой информации

   - f(x)
   - f(x), f'(x) (требуется вычисление и ручное программирование производных)
   - ...

   Если $x \in \mathbb{R}^n$, то надо вычислять производные по всем n координатам

2) - если выдает такое x*, что $\nabla f(x^*) = 0$, то это стационарная точка первого рода
   - если выдает такое x*, что $\nabla f(x^*) = 0$ и $(Hh, h) > 0$, то это стационарная точка второго рода и мы точно можем сказать, что она не седловая

3) Согласно решаемой задаче:

   - Находит условный экстремум
   - Находит безусловный экстремум

4) - Детерменированные – алгоритмический процесс, который выдаёт уникальный и предопределённый результат для заданных входных данных
   - Стахостические – алгоритм дает программу решения задачи несколькими путями, приводящими к вероятностному достижению результата

Классификация алгоритмов зеркальна классификации задач оптимизации. Лучшим решением является наиболее специализированное. Нет универсального алгоритма, который работал бы при всех условиях.

Рассматриваем алгоритмы, которые гарантируют сходимость к какой-то точке: $x_n \mapsto x^*$

1) Линейная сходимость

   $\|x_{n+1} - x^*\| \leq q\|x_n - x^*\|$

   $q \in [0, 1]$, $\forall \, n \geq n_0$

2) Квадратичная сходимость(лучше линейной)

   $\|x_{n+1} - x^*\| \leq c\|x_n - x^*\|^2$

   $\exists \, c, n_o : \, \forall n \geq n_0$

3) Сверхлинейная сходимость

$\|x_{n+1} - x^*\| \leq q_n \|x_n - x^*\|,$

где $q_n \mapsto 0$, при $n \mapsto \inf$

Вводим последовательность $\underline{x_{n+1}} = \underline{x_n} - \alpha_n \nabla f(\underline{x_n})$, где $\alpha_n > 0$
$x_n \mapsto x^*$
$\nabla f(x^*) = 0$
Разложим функцию в ряд Тейлора:
$f(\underline{x}) = f(\underline{x_n}) + (\nabla f(x_n), \underline{x} - \underline{x_n}) + \frac{1}{2}(\underline{x} - \underline{x_n}, H(\underline{x} - \underline{x_n})) \ \mapsto min,$
где H - матрица Гесса.
Это задача квадратичного программирования.
Можем выбрать x_{n+1} так, чтобы локальное разложение было минимальным. Его можно вычислить аналитически.
$x_{n+1} = argmin($**разложения(хз, что это все значит)**$)$
При этом все может сломаться, если разложение будет справедливо, но функция будет иметь очень большой рост.
Пусть у нас есть функция $f(\underline{x})$ и мы хотим найти ее минимум(максимум).
Такая функция называется целевой (cost function).
Кроме того имеется набор ограничений, которые заданы в виде:
$\begin{cases} g_i(x) = 0, \ i \in \varepsilon \\ g_i(x) \geq 0, \ i \in I \end{cases}$
Это есть задача математического программирования - ограниченная задача оптимизации(constrained
Система уравнений (неравенств) на g определяет множества X, которое назвается допустимым (feasible) множетвом.
Множество $A(x) = \{i \in \varepsilon \cup I | g_i(x) = 0\}$ назывется множеством активных ограничений.
Те i, которые не принадлежат ему($i \notin A(x)$), составляют множество пассивных ограничений.
Из написанного выше видно, что для каждой точки x это какие-то свои множества.

Теорема о необходимом условии существования локального решения при наличии ограничений[6]

Пусть функции f($\underline{x}$) и $g_i(x)$ - непрерывно дифференцируемы хотя бы в какой-то окрестность точки x* и пусть x* - локальное решение. Тогда $\exists$ $\underline{y^*}$ - вектор множителей лагранжа - такой, что:

1) $\nabla_x f(\underline{x}) - \sum y_i * \nabla_x g_i(\underline{x}) = 0 = L(\underline{x}, \underline{y})$

2) $g_i(\underline{x^*}) = 0$ при $i \in \varepsilon$

3) $g_i(\underline{x^*}) \geq 0$ и $y_i^* \geq 0$ при $i \in I$

4) $g_i(\underline{x^*}) * y_i = 0$ - условие дополняющей нежесткости

При $g_i(\underline{x^*}) > 0$ соответствующий множитель Лагранжа равен 0.
Этот набор условий называется условиями Каруша-Куна-Такера
Контрпример:
$f(\underline{x}) = x_1$, $\underline{x} \in \ \mathbb{R}^2$
Условия:
$x_1^2 - x_2 \geq 0$

[6]для регулярных задач

$x_1^3 + x_2 \geq 0$
$1 - x_1^2 - x_2^2 \geq 0$
Минимум функции f(x) достигается в точке (0, 0)
Посчитаем все градиенты ∇_x функции и ограничений:
(1, 0)
(0, 1)
(0, 1)
(0, 0)
У нас должно быть по условию теоремы:
$\begin{pmatrix}1\\0\end{pmatrix} -y_1 \begin{pmatrix}0\\-1\end{pmatrix} -y_2 \begin{pmatrix}0\\1\end{pmatrix} -y_3 \begin{pmatrix}0\\0\end{pmatrix} = \begin{pmatrix}0\\0\end{pmatrix}$, что не может выполняться.

Если x* - локальное решение, то $\nabla_x f$ - линейная комбинация $\nabla g_i(x)$ $i \in A(x)$, т.е. градиент в данной точке является линейной комбинацией градиентов всех активных решений.

Введем пространство N из векторов, ортогональных всем градиентам ограничений:
N : $(s_1, \nabla g_i(x))$, $i \in A(x)$
$(s, \nabla f(x)) = 0$, $\forall s \in N$

Если бы у нас имелась проекция градиента в этом направлении, то это означает, что без нарушения ограничений мы могли бы отступить вдоль этого вектора и наша целевая функция бы уменьшилась, что противоречит условию локального минимума. Тогда необходимое условие экстремума 1-ого порядка: проекция градиента целевой функции на нуль-пространство(N) равна 0, т.е. получаем аналогию к задаче без ограничений: проекция градиента на какое-то подпространство равна 0. Т.е. в направлении, ортогональном направлению активных ограничений, градиент равен 0, а в подпространстве, где ограничения образуют базис, он выражается как линейная комбинация всех градиентов ограничений.

Необходимые условия для экстремумов 2 порядка:

f(x) и g(x) дважды непрерывно дифференцируемы, и x* - локальный минимум, тогда существет такой вектор множителей Лагранжа $\underline{y}^*$, что выполняются все условия из предыдущей теоремы, но еще и
$(\underline{h}, \frac{\partial^2 L}{\partial x_i \partial x_j}) \geq 0$
для $\underline{h} \in N_+$, где $(\underline{h}, \nabla_x g_i(x)) \geq 0$, при $i \in A(x^*) \cup I$, $y_i^* = 0$

т.е. направление $\underline{h}$ такое, что он составляет острый угол с направлениями градиентов активных ограничений (для активных неравенств, для которых вектор множителей Лагранжа равен 0)

Можно ввести теорему про достаточное условие, но там будет строго положительно определенная матрица вторых производных.

Самое важное!

Если f(x) - выпуклое и X(множество ограничений) - выпуклое, то все локальные решения становятся глобальными

2 способа синтезировать алгоритмы задач с ограничениями, используя готовые(существующие) подходы:

1) Метод барьерных (штрафных) функций

2) Метод проекций градиента

Их нельзя заменить друг на друга.
Возможно 2 варианта для X - области определения функции f(x):

- Она определена на хорошем или большом можестве ($\mathbb{R}^n$), но мы делаем какие-то разумные ограничения для параметров (например, физические)

- Область определения не совпадает со всем пространством, т.е. есть области, где мы не можем вычислить эту функцию

X принадлежит какому-то компактному (ограниченному и замкнутому множеству), $x \in \underline{P} \subset \mathbb{R}^n$, или можно сказать, что совпадает с $\mathbb{R}$

Представим, что числа с плавающей точкай могут хранить какой-то ограниченный набор значений (число с плавающей точкой двойной точностью не больше 10^{358})

$f(x) \mapsto min,\ g_i(x) \geq x,\ i \in I$

Представим, что ограничений в виде равенств нет, т.к. их можно разрешить как систему нелинейных алгебраических уравнений и подставить в целевую функцию.

Предлагается найти какой-то функцию $\varphi(\underline{x}, C)$, которая является барьерной или штрафной.

Для нее выполняются следующие условия:

1) $\varphi(\underline{x}, C) \geq 0,\ C \geq 0$ при $x \in \underline{P} \subset \mathbb{R}^n$(т.е. всех x, которые нас интересуют)

2) Имеет разные ассимпототики на $\pm\infty$:

   - $\varphi(\underline{x}, C) \mapsto 0$, при $C \mapsto +\infty$ на $\underline{x} \in X$, т.е. на всех x из области допустимых значений
   - $\varphi(\underline{x}, C) \mapsto \infty$, при $C \mapsto +\infty$ на $\underline{x} \notin X$, т.е. на всех x вне области допустимых значений

Примеры:

$\varphi(\underline{x}, C) = \sum_i exp(-Cg_i(x))$

$\varphi(\underline{x}, C) = C \sum min(0, g_i(x))$

Зачем это надо?

Заменим f на $\Phi(\underline{x}) = f(\underline{x}) + \varphi(\underline{x}, C)$ - ищем минимум этой функции

1) $\min\limits_{x \in X} f(\underline{x}) = \lim\limits_{C \mapsto \infty} \inf\limits_{x \in P \subset \mathbb{R}^n} \Phi(\underline{x}, C)$

   т.е. минимум искомой функции - предел точных нижней граней $\Phi(\underline{x}, C)$

2) Если существует последовательность x_k, причем для нее

   $\Phi(\underline{x_k}, C_k) \leq inf\ \Phi(x_k, C) - \varepsilon_k$

   x_k - решение задачи оптимизации Φ с точностью ε

   Если $C_k \mapsto \infty,\ \varepsilon_k \mapsto \infty$, то $x_k \mapsto x^*$

Замечание:

Нельзя сразу подставлять большое C, т.к. могут возникнуть проблемы, например, с округлением.

Проблема:

В какой-то момент может потребоваться вычислить нашу целевую функцию за пределами допустимого множестваю

Важно помнить про классификацию:

- одни алгоритмы всегда остаются в пределах допустимой области
- Другие же могут выходить за ее пределы.

$\underline{a} \in \mathbb{R}^n,\ X,\ \pi_x(\underline{a})-$ проекция a на множество X

Проекцией точки a на X называется такое $x \in X$, что $\|x - a\|^2 \mapsto \min\limits_{x \in X}$, т.е. такой элемент, который по норме ближе всех

- $\pi_x(\underline{a}) = x_0 + \frac{\underline{a}-\underline{x_0}}{\|\underline{a}-\underline{x_0}\|}R$ - проекция точки для шара
- Если есть условие, что $x_i \geq 0$

  $\pi_x(a) = \begin{pmatrix} max(0, a_1) \\ ... \\ max(0, a_N) \end{pmatrix}$
- $A\underline{x} = b$

  $\pi_x(\underline{a}) = \underline{a} - A^T(AA^{-1})^{-1}(A\underline{a} - \underline{b})$ - для афинного множества

Сколько будет проекций центра шара на внешнее множество?
Рассмотрим X -замкнутое и выпуклое

1) Существует единственная проекция $\pi_x(a)$
2) $(\pi_x(\underline{a}) - \underline{a}, x - \pi_x(\underline{a})) \geq 0$, т.е. направления образуют острые углы
3) $\|\pi_x(\underline{a_1}) - \pi_x(\underline{a_2})\| \leq \|a_1 - a_2\|, \ \forall a_1, a_2$

   При проектировании пары точек расстяние между ними не увеличивается.

Если все эти условия выполнены и f(x) - выпуклая, тогда необходимое и достаточное условие решения задачи:
$\underline{x^*}$ - минимум, если:
$x^* = \pi_x(x^* - \alpha\nabla_x f(x^*))$, при $\alpha > 0$
Проекция, из которой вычтен антиградиент.
Т.е. мы отошли от какой-то точки, потом взяли проекцию и оказались в ней же. Оператор проекции уничтожил добавку.
Во второе условие подставляем x* вместо a, а вместо проекции вторую часть уравнения:
$(\underline{x^*} - (\underline{x^*} - \alpha\nabla_x f(x^*), \underline{x} - \underline{x^*}) \geq 0$
$\alpha(\nabla_x f(x^*), x - x^*) \geq 0$
Самый простой способ решения задачи оптимизации с выпуклым множеством X
Модифицируем метод градиентного спуска
Было: $\underline{x_{n+1}} = \underline{x_n} - \alpha_n\nabla_x f(\underline{x_n})$
Стало: $\underline{x_{n+1}} = \pi_x(\underline{x_n} - \alpha_n\nabla_x f(\underline{x_n}))$ - она действительно сходится.
Пример
Алгоритм для решения частной задачи.
Хотим минимизировать $\|Ax - b\|^2$, так, чтобы $x_i \geq 0, \ \forall i$ Рисуем уровни одинакового значения.
Не можем просто взять решение без ограничений ($x = (A^*A)^{-1}A^Tb$) и занулить нужные x, т.к. получим не точку с наименьшей возможной разностью норм.
Можем рассмотреть множества $X = 0$, $X_1 = 0$, $X_2 = 0$, $X_1, X_2 \neq 0$.
Подставляем все 4 варианта, ищем, где условия выполнены, и выбираем минимальное по норме значение. Но для n-мерного пространства количество граней(границ недокубика (положительного ортанта)) возратает как 2^n. То есть мы берем все возможные комбинации и их 2^n. Хотим более умно перебирать грани допустимого множества.
Грани сопоставляем с множеством Z - множеством индексов - Z: i: $X_i = 0$
$P : i \notin Z, X_i > 0$
Строим последовательность действий, получающую получить из текущего $Z^k \to Z^{k+1}$ и $X^k \to X^{k+1}$
$\underline{x} = 0$ - начало, при этом Z = 1, 2, ..., N

Машинное обучение

Билет 1. Основная идея подхода к решению задач методами машинного обучения. Фундаментальное ограничение машинного обучения.

Машинное обучение – обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. Машинное обучение находится на стыке математической статистики, методов оптимизации и классических математических дисциплин, но имеет также и собственную специфику, связанную с проблемами вычислительной эффективности и переобучения. Многие методы разрабатывались как альтернатива классическим статистическим подходам.

Обучение по прецедентам, или индуктивное обучение, основано на выявлении общих закономерностей по частным эмпирическим данным. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний. Дедуктивное обучение принято относить к области экспертных систем, поэтому термины машинное обучение и обучение по прецедентам можно считать синонимами.

Дано конечное множество прецедентов (объектов, ситуаций), по каждому из которых собраны (измерены) некоторые данные. Данные о прецеденте называют также его описанием. Совокупность всех имеющихся описаний прецедентов называется обучающей выборкой. Требуется по этим частным данным выявить общие зависимости, закономерности, взаимосвязи, присущие не только этой конкретной выборке, но вообще всем прецедентам, в том числе тем, которые ещё не наблюдались.

Наиболее распространённым способом описания прецедентов является признаковое описание. Фиксируется совокупность n показателей, измеряемых у всех прецедентов. Если все n показателей числовые, то признаковые описания представляют собой числовые векторы размерности n. Возможны и более сложные случаи, когда прецеденты описываются временными рядами или сигналами, изображениями, видеорядами, текстами, попарными отношениями сходства или интенсивности взаимодействия, и т. д.

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

Выходом алгоритма обучения является функция, аппроксимирующая неизвестную (восстанавливаемую) зависимость. В задачах классификации аппроксимирующую функцию принято называть классификатором (classifier), концептом (concept) или гипотезой (hypothesys); в задачах восстановления регрессии — функцией регрессии; иногда просто функцией. В русскоязычной литературе аппроксимирующую функцию также называют алгоритмом, подчёркивая, что и она должна допускать эффективную компьютерную реализацию.

Практически всё обучение, которое мы ждём от компьютеров, состоит в сокращении информации до основных закономерностей, на основании которых можно делать выводы о чём-то неизвестном. Рассмотрим загадку:

$$1 + 4 = 5$$
$$2 + 5 = 12$$
$$3 + 6 = 21$$
$$8 + 11 = ?$$

Достаточно нетрудно заметить две закономерности:

$$1 * (4 + 1) = 5$$
$$2 * (5 + 1) = 12$$
$$3 * (6 + 1) = 21$$

И:

$$0 + 1 + 4 = 5$$
$$5 + 2 + 5 = 12$$
$$12 + 3 + 6 = 21$$

Какая же закономерность верна? Естественно, обе – и ни одна из них. Всё зависит от того, какие закономерности допустимы. Поиск закономерностей зависит от предположений наблюдателя.

То же верно и для МО. Даже когда машины обучают сами себя, предпочтительные закономерности выбираются людьми: должно ли ПО для распознавания лиц содержать явные правила если/то, или оно должно расценивать каждую особенность как дополнительное доказательство в пользу или против каждого возможного человека, которому принадлежит лицо? Какие особенности изображения должно обрабатывать ПО? Нужно ли ей работать с отдельными пикселями? Выбор подобных вариантов ограничивает то, какие закономерности система сочтёт вероятными или даже возможными. Эти вопросы ограничивают попытки применения нейросетей к новым задачам.

Билет 2-3, 9. Классификация методов машинного обучения. Приведите примеры задач каждого класса. Задача классификации с учителем. Примеры. Методы машинного обучения без учителя. Приведите примеры задач.

Обучение с учителем (supervised learning) — наиболее распространённый случай. Каждый прецедент представляет собой пару «объект, ответ». Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Функционал качества обычно определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки.

- Задача классификации (classification) отличается тем, что множество допустимых ответов конечно. Их называют метками классов (class label). Класс — это множество всех объектов с данным значением метки. Примеры: классификация цветов (датасет про ирисы), классификация частиц в последнем домашнем задании 2021 учебного года, классификация клиентов на платёжеспособных и нет.

- Задача регрессии (regression) отличается тем, что допустимым ответом является действительное число или числовой вектор. Примеры: стоимость квартиры, красное смещение в домашнем задании 5 2021 учебного года.

- Задача ранжирования (learning to rank) отличается тем, что ответы надо получить сразу на множестве объектов, после чего отсортировать их по значениям ответов. Может сводиться к задачам классификации или регрессии. Пример: ранжирование страниц в поиске в Интернете.

- Задача прогнозирования (forecasting) отличается тем, что объектами являются отрезки временных рядов, обрывающиеся в тот момент, когда требуется сделать прогноз на будущее. Пример: прогноз спроса на товар.

Обучение без учителя (unsupervised learning). В этом случае ответы не задаются, и требуется искать зависимости между объектами.

- Задача кластеризации (clustering) заключается в том, чтобы сгруппировать объекты в кластеры, используя данные о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний. Пример: позитронно-эмиссионная томография для автоматического выделения различных типов тканей на трехмерном изображении.

- Задача поиска ассоциативных правил (association rules learning). Исходные данные представляются в виде признаковых описаний. Требуется найти такие наборы признаков, и такие значения этих признаков, которые особенно часто (неслучайно часто) встречаются в признаковых описаниях объектов. Пример: поиск ассоциативных правил в заданном наборе транзакций (хлеб, молоко и так далее).

- Задача фильтрации выбросов (outliers detection) — обнаружение в обучающей выборке небольшого числа нетипичных объектов. В некоторых приложениях их поиск является самоцелью. В других приложениях эти объекты являются следствием ошибок в данных или неточности модели, то есть шумом, мешающим настраивать модель, и должны быть удалены из выборки. Пример: обнаружение мошенничества.

- Задача построения доверительной области (quantile estimation) — области минимального объёма с достаточно гладкой границей, содержащей заданную долю выборки.

- Задача сокращения размерности (dimensionality reduction) заключается в том, чтобы по исходным признакам с помощью некоторых функций преобразования перейти к наименьшему числу новых признаков, не потеряв при этом никакой существенной информации об объектах выборки. В классе линейных преобразований наиболее известным примером является метод главных компонент.

- Задача заполнения пропущенных значений (missing values) — замена недостающих значений в матрице объекты–признаки их прогнозными значениями.

Это были главные, но ради интереса стоит указать:

Частичное обучение (semi-supervised learning) занимает промежуточное положение между обучением с учителем и без учителя. Каждый прецедент представляет собой пару «объект, ответ», но ответы известны только на части прецедентов. Пример: автоматическая рубрикация большого количества текстов при условии, что некоторые из них уже отнесены к каким-то рубрикам.

Обучение с подкреплением (reinforcement learning). Роль объектов играют пары «ситуация, принятое решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды). Как и в задачах прогнозирования, здесь существенную роль играет фактор времени. Примеры: формирование инвестиционных стратегий, автоматическое управление технологическими процессами, самообучение роботов.

Динамическое обучение (online learning) может быть как обучением с учителем, так и без учителя. Специфика в том, что прецеденты поступают потоком. Требуется немедленно принимать решение по каждому прецеденту и одновременно доучивать модель зависимости с учётом новых прецедентов. Как и в задачах прогнозирования, здесь существенную роль играет фактор времени. Пример: системы автопилотов.

Активное обучение (active learning) отличается тем, что обучаемый имеет возможность самостоятельно назначать следующий прецедент, который станет известен.

Метаобучение (meta-learning или learning-to-learn) отличается тем, что прецедентами являются ранее решённые задачи обучения. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель — обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

Многозадачное обучение (multi-task learning). Набор взаимосвязанных или схожих задач обучения решается одновременно, с помощью различных алгоритмов обучения, имеющих схожее внутренне представление. Информация о сходстве задач между собой позволяет более эффективно совершенствовать алгоритм обучения и повышать качество решения основной задачи.

Индуктивный перенос (inductive transfer). Опыт решения отдельных частных задач обучения по прецедентам переносится на решение последующих частных задач обучения. Для формализации и сохранения этого опыта применяются реляционные или иерархические структуры представления знаний.

Глубинное обучение может проходить как без учителя, так и с подкреплением. При глубинном обучении частично имитируются принципы обучения людей — используются нейронные сети для все более подробного уточнения характеристик набора данных. Глубинные нейронные сети применяются, в частности, для ускорения скрининга больших объемов данных при поиске лекарственных средств. Такие нейросети способны обрабатывать множество изображений за короткое время и извлечь больше признаков, которые модель в конечном счете запоминает. Глубинное обучение может использоваться в автомобильной отрасли при выполнении ремонта и профилактического обслуживания.

Билет 4. Функция потерь в задачах машинного обучения.

Предположим, дано задание наполнить мешок 5 кг муки. Вы заполняете его до тех пор, пока измерительный прибор не даст идеальное показание 5 кг, или вы достанете песок, если показание превысит 5 кг.

Точно так же, как весы, если ваши прогнозы не верны, ваша функция потерь будет выводить большее число. Если они довольно хороши, выведите меньшее число. Когда вы экспериментируете с вашим алгоритмом, чтобы попытаться улучшить свою модель, ваша функция потерь скажет вам, достигаете ли вы (или достигаете) где-либо.

Функция, которую мы хотим минимизировать или максимизировать, называется целевой функцией или критерием. Когда мы минимизируем его, мы можем также назвать его функцией стоимости, функцией потерь или функцией ошибки.

По своей сути функция потерь - это мера того, насколько хороша ваша модель прогнозирования с точки зрения возможности прогнозировать ожидаемый результат (или значение). Мы преобразуем задачу обучения в задачу оптимизации, определяем функцию потерь, а затем оптимизируем алгоритм, чтобы минимизировать функцию потерь.

Примеры:

- Mean Squared Error (MSE) - это рабочая область базовых функций потерь, так как она проста для понимания и реализации и в целом работает довольно хорошо. Чтобы рассчитать MSE, вы берете разницу между предсказаниями вашей модели и основополагающей правдой, вычеркиваете ее и затем усредняете по всему набору данных. Результат всегда положительный, независимо от знака предсказанных и основанных значений истинности, и идеальное значение равно 0,0.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

- Mean Absolute Error (MAE) - лишь немного отличается по определению от MSE, но, что интересно, обеспечивает почти совершенно противоположные свойства. Чтобы рассчитать MAE, вы берете разницу между предсказаниями вашей модели и основополагающей правдой, применяете абсолютное значение к этой разнице, а затем усредняете его по всему набору данных.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

- Функция потерь Хьюбера — это функция потерь, используемая в устойчивой регрессии, которая менее чувствительна к выбросам, чем квадратичная ошибка.

$$\text{HuberLoss} = \begin{cases} \frac{1}{2}a^2 & \text{для } |a| \le \delta, \\ \delta\left(|a| - \frac{1}{2}\delta\right), & \text{иначе.} \end{cases} \quad (1)$$

- Функция потерь для классификации - индикатор ошибки.

$$\text{ClassificationLoss} = \sum_{i=1}^{n} [a(x)_i \neq y(x)_i]$$

Билет 5. Понятие переобучения. Тестовая выборка.

Тестовая (или контрольная) выборка (test sample) — выборка, по которой оценивается качество построенной модели. Если обучающая и тестовая выборки независимы, то оценка, сделанная по тестовой выборке, является несмещённой.

Оценку качества, сделанную по тестовой выборке, можно применить для выбора наилучшей модели. Однако тогда она снова окажется оптимистически смещённой. Для получения немсещённой оценки выбранной модели приходится выделять третью выборку.

Переобучение, переподгонка (overtraining, overfitting) — нежелательное явление, возникающее при решении задач обучения по прецедентам, когда вероятность ошибки обученного алгоритма на объектах тестовой выборки оказывается существенно выше, чем средняя ошибка на обучающей выборке. Переобучение возникает при использовании избыточно сложных моделей.

Билет 6. Идея метода k ближайших соседей (kNN).

Метод ближайших соседей (kNN - k Nearest Neighbours) - метод решения задач классификации и задач регрессии, основанный на поиске ближайших объектов с известными значения целевой переменной. Метод основан на предположении о том, что близким объектам в признаковом пространстве соответствуют похожие метки. Для нового объекта метод предполагает найти ближайшие к нему объекты и построить прогноз по их меткам.

В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны. В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Алгоритм может быть применим к выборкам с большим количеством атрибутов (многомерным). Для этого перед применением нужно определить функцию расстояния; классический вариант такой функции — евклидова метрика.

Разные атрибуты могут иметь разный диапазон представленных значений в выборке (например атрибут А представлен в диапазоне от 0.1 до 0.5, а атрибут Б представлен в диапазоне

от 1000 до 5000), то значения дистанции могут сильно зависеть от атрибутов с большими диапазонами. Поэтому данные обычно подлежат нормализации. Некоторые значимые атрибуты могут быть важнее остальных, поэтому для каждого атрибута может быть задан в соответствие определённый вес (например вычисленный с помощью тестовой выборки и оптимизации ошибки отклонения). При взвешенном способе во внимание принимается не только количество попавших в область определённых классов, но и их удалённость от нового значения.

Гиперпараметры - это настраиваемые параметры, которые необходимо настроить, чтобы получить модель с оптимальными характеристиками. В случае kNN таковым является параметр k - числа соседей.

Билет 7. Гиперпараметры и валидационная выборка.

Гиперпараметры модели — параметры, значения которых задается до начала обучения модели и не изменяется в процессе обучения. У модели может не быть гиперпараметров.

Параметры модели — параметры, которые изменяются и оптимизируются в процессе обучения модели и итоговые значения этих параметров являются результатом обучения модели.

Примерами гиперпараметров могут служить количество слоев нейронной сети, а также количество нейронов на каждом слое. Примерами параметров могут служить веса ребер нейронной сети. Для нахождения оптимальных гиперпараметров модели могут применяться различные алгоритмы настройки гиперпараметров.

Примеры: число деревьев в случайном лесе, параметр k для метода kNN, глубина деревьев в бэггинге.

Валидационная выборка (validation sample) — выборка, по которой осуществляется выбор наилучшей модели из множества моделей, построенных по обучающей выборке. Её объём может быть равен, например 0.1 от общего.

Билет 8. Задача регрессии с учителем. Примеры. Популярные алгоритмы решения.

Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число (2, 35, 76.454 и др.), к примеру цена квартиры, стоимость ценной бумаги по прошествии полугода, ожидаемый доход магазина на следующий месяц, качество вина при слепом тестировании. Популярные алгоритмы решения:

- Линейная и полиномиальная регрессия

  Начнём с простого. Одномерная (простая) линейная регрессия – это метод, используемый для моделирования отношений между одной независимой входной переменной (переменной функции) и выходной зависимой переменной. Модель линейная.

  Более общий случай – множественная линейная регрессия, где создаётся модель взаимосвязи между несколькими входными переменными и выходной зависимой переменной. Модель остаётся линейной, поскольку выходное значение представляет собой линейную комбинацию входных значений.

  Также стоит упомянуть полиномиальную регрессию. Модель становится нелинейной комбинацией входных переменных, т. е. среди них могут быть экспоненциальные переменные: синус, косинус и т. п. Модели регрессии можно обучить с помощью метода стохастического градиента.

- Дерево принятия решений и Случайный лес

  Начнём с простого случая. Дерево принятия решений – это представления правил, находящихся в последовательной, иерархической структуре, где каждому объекту соответствует узел, дающий решение. При построении дерева важно классифицировать атрибуты так, чтобы создать “чистые” узлы. То есть выбранный атрибут должен разбить множество так, чтобы получаемые в итоге подмножества состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому, т.е. количество объектов из других классов в каждом из этих множеств было как можно меньше.

  “Случайный лес” – совокупность деревьев принятия решений. Входной вектор проходит через несколько деревьев решений. Для регрессии выходное значение всех деревьев усредняется; для классификации используется схема голосования для определения конечного класса.

- Нейронные сети

  Нейронная сеть состоит из взаимосвязанных групп узлов, называемых нейронами. Входные данные передаются в эти нейроны в виде линейной комбинации со множеством переменных. Значение, умножаемое на каждую функциональную переменную, называется весом. Затем к этой линейной комбинации применяется нелинейность, что даёт нейронной сети возможность моделировать сложные нелинейные отношения. Чаще всего нейросети бывают многослойными: выход одного слоя передается следующему так, как описано выше. На выходе нелинейность не применяется.

Билет 10. Идея метода нейронных сетей.

Нейронная сеть – математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

НС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Основная мысль позаимствована у природы: есть связанные между собой нейроны, которые передают друг другу сигналы. Есть сеть из нейронов, соединённых между собой. Нейроны возбуждаются под действием входов и передают возбуждение (либо как один бит, либо с каким-то

значением) дальше. В результате последний нейрон на выход подаёт ответ. Нейрон возбуждается, если выполнено некоторое условие на входах. Затем он передаёт свой импульс дальше.

Нейрон возбуждается под действием какой-то функции от входов. Такая конструкция называется перцептроном. Это простейшая модель нейрона в искусственной нейронной сети. У линейного перцептрона заданы: n весов; лимит активации; выход перцептрона o вычисляется так: 1, если сумма весов и признаков >0 и -1 иначе.

Один перцептрон может реализовать любую гиперплоскость, рассекающую пространство возможных решений. Иначе говоря, если прообразы 0 и 1 у целевой функции линейно отделимы, то одного перцептрона достаточно. Но он не может реализовать линейно неотделимое множество решений, например, XOR.

Всё, что может у перцептрона меняться – это веса. Их мы и будем подправлять при обучении. Если перцептрон отработал правильно, веса не меняются. Если неправильно – сдвигаются в нужную сторону. Перцептроны могут быть не только линейными, но для идеи метода хватит и этого.

Также можно сказать про:

Метод обратного распространения ошибки (англ. backpropagation) – метод вычисления градиента, который используется при обновлении весов многослойного перцептрона. Основная идея этого метода состоит в распространении сигналов ошибки от выходов сети к её входам, в направлении обратном прямому распространению сигналов в обычном режиме работы.

Билет 11. Идея методов random forest и gradient boosting.

Random forest – алгоритм машинного обучения, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга, и метод случайных подпространств. Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

Все деревья строятся независимо по следующей схеме:

Выбирается подвыборка обучающей выборки размера samplesize – по ней строится дерево (для каждого дерева — своя подвыборка). Для построения каждого расщепления в дереве просматриваем maxfeatures случайных признаков (для каждого нового расщепления – свои случайные признаки). Выбираем наилучшие признак и расщепление по нему (по заранее заданному критерию). Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса), но в современных реализациях есть параметры, которые ограничивают высоту дерева, число объектов в листьях и число объектов в подвыборке, при котором проводится расщепление.

Достоинства:

- Способность эффективно обрабатывать данные с большим числом признаков и классов.
- Нечувствительность к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков.
- Одинаково хорошо обрабатываются как непрерывные, так и дискретные признаки. Существуют методы построения деревьев по данным с пропущенными значениями признаков.
- Существуют методы оценивания значимости отдельных признаков в модели.
- Высокая параллелизуемость и масштабируемость.

Основной недостаток: Большой размер получающихся моделей. Требуется $O(K)$ памяти для хранения модели, где K – число деревьев.

Другим методом улучшения предсказаний является бустинг (boosting), идея которого заключается в итеративном процессе последовательного построения частных моделей. Каждая новая модель обучается с использованием информации об ошибках, сделанных на предыдущем этапе, а результирующая функция представляет собой линейную комбинацию всего ансамбля моделей с учетом минимизации любой штрафной функции. Подобно бэггингу, бустинг является общим подходом, который можно применять ко многим статистическим методам регрессии и классификации.

Бутстреп-выборки в ходе реализации бустинга не создаются, но вместо этого каждое дерево строится по набору данных X, r который на каждом шаге модифицируется определенным образом. На первой итерации по значениям исходных предикторов строится дерево $f_1(x)$ и находится вектор остатков r_1. На последующем этапе новое регрессионное дерево $f_2(x)$ строится уже не по обучающим данным X, а по остаткам r_1 предыдущей модели. Линейная комбинация прогноза по построенным деревьям дает нам новые остатки r_2 ,$r_1 + \lambda f_2(x)$, и этот итерационный процесс повторяется Z раз. Благодаря построению неглубоких деревьев по остаткам, прогноз отклика медленно улучшается в областях, где одиночное дерево работает не очень хорошо. Такие деревья могут быть довольно небольшими, лишь с несколькими конечными узлами. Параметр сжатия λ регулирует скорость этого процесса, позволяя создавать комбинации деревьев более сложной формы для “атаки” остатков. Итоговая модель бустинга представляет собой ансамбль.

Билет 12. Приведите примеры физических задач для которых подходит и не подходит машинное обучение.