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 **2**

Структуры данных **3**

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. 3

Билет 4. Сортировка. Приведите примеры методов сортировки. 4

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. 4

Оптимизация **6**

Билет 1. Задачи оптимизации. Их классификация. Примеры 7

Машинное обучение **7**

Билет 1. Основная идея подхода к решению задач методами машинного обучения. Фундаментальное ограничение машинного обучения. 7

Билет 2-3, 9. Классификация методов машинного обучения. Приведите примеры задач каждого класса. Задача классификации с учителем. Примеры. 8

Билет 4. Функция потерь в задачах машинного обучения. 10

Билет 5. Понятие переобучения. Тестовая выборка. 11

Билет 6. Идея метода k ближайших соседей (kNN). 11

Билет 7. Гиперпараметры и валидационная выборка. 12

Билет 8. Задача регрессии с учителем. Примеры. Популярные алгоритмы решения. . . 12

Билет 10. Идея метода нейронных сетей. 13

Билет 11. Идея методов random forest и gradient boosting. 14

Билет 12. Приведите примеры физических задач для которых подходит и не подходит машинное обучение. 15

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.

Оптимизация

Билет 1. Задачи оптимизации. Их классификация. Примеры

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.

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.

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

- 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.

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

- 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 $r_2 \text{¸} r_1 + \lambda f_2(x)$, 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.