

Машинное обучение

Гаврилов Андрей Владимирович

К.т.н., доцент

НГТУ, кафедра вычислительной техники

Email: gavrilov@corp.nstu.ru

<http://www.ermak.cs.nstu.ru/neurotech>

Цели и задачи дисциплины

- Освоение разработки и применения методов статистического распознавания образов в машинном обучении.
- Формирование знаний, умений и навыков постановки задач по разработке или совершенствованию методов и алгоритмов машинного обучения для решения задач искусственного интеллекта

Из рабочей программы СПбЭТУ.

Содержание дисциплины

- Лекции (18 часов)
- 8 2-часовых практических занятий в 4-х лабораторных работах (по 2 в каждой) по методическим указаниям Н.А.Жуковой (СпбГЭТУ «ЛЭТИ»)
- Контрольная работа
- Экзамен в виде теста или собеседования
- Баллы:
 - За каждую сданную практическую работу 0-6 баллов, максимум – 48 баллов
 - Контрольная работа – 0-12
 - Экзамен – 0-40 баллов

Лекции

- Лекция 1. Введение. Основные понятия и методы машинного обучения. Обучение с подкреплением
- Лекция 2. Основные модели нейронных сетей для обучения с учителем
- Лекция 3. Основные модели нейронных сетей для обучения без учителя
- Лекция 4. Глубокие нейронные сети
- Лекция 5. Обработка естественного языка нейронными сетями
- Лекция 6. Генетические алгоритмы
- Лекция 7. Методика использования машинного обучения
- Дополнительные лекции

Возможные темы контрольной работы

- Если тема МД как-то связана с ИИ:
 - Обзор литературы по теме МД
 - Расширенная постановка задачи МД
- Разработка программы, демонстрирующей какой-либо метод МО (не нейронные сети).
Согласовать с преподавателем.
Оформляется наподобие отчета по лабораторной работе с небольшой методической частью.
- Перевод статьи и краткая презентация по ней на рус. языке
Из перечня <https://disk.yandex.ru/d/9xIwDGjF23ONKw>
Можете сами предложить статью для перевода.

- Материалы (лекции, методические материалы по лабораторным работам, некоторая литература и экзаменационные вопросы) на Dispace 2.0

<http://dispace.edu.nstu.ru>

Дисциплина «Машинное обучение»,
автор – А.В.Гаврилов,

Введение

Arthur Samuel (1959, автор первой обучающейся программы, играющей в шашки), Machine learning - “Поле исследования, которое дает возможность компьютеру обучаться не будучи явно запрограммированным”.

Tom Mitchell (1998):

“ Компьютерная программа перенимает опыт E в отношении задачи T и некоторого показателя измерения P , если ее эффективность на T , измеренная по P улучшается с опытом E ”.

Definition: A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

Из книги T.Mitchel. Machine Learning. - McGraw-Hill Science, 1997.

Определение

- **Машинное обучение** (*machine learning*, ML) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме. (из Википедии)

Определение (2)

- Термин машинное обучение охватывает большой круг технологий, которые преследуют одну цель: извлечь имеющую смысл информацию из данных.
- Машинное обучение заключается в извлечении знаний из данных. Это научная область, находящаяся на пересечении статистики, искусственного интеллекта и компьютерных наук и также известная как прогнозная аналитика или статистическое обучение.

Из конспекта лекций Н.А.Жуковой (СПбЭТУ)

Определение (3)

- Машинное обучение – набор методов, алгоритмов и технологий, позволяющих разрабатывать системы искусственного интеллекта, способные обучаться или самообучаться с целью решения определенного круга задач.

Применение машинного обучения

- Обработка запросов в поисковых системах
- Распознавание фотографий
- Распознавание рукописного текста
- Фильтры почтового спама
- Интеллектуальный анализ баз данных (лучшее обслуживание потребителей, вычислительная биология, машиностроение,...)
- Автономное управление машинами (автомобили, вертолеты,...)
- Выдавать индивидуальные рекомендации - подстраиваться под ваши предпочтения
- Генерация правдоподобных текстов и изображений
- И т.д. и т.п.

Место машинного обучения в ИИ



Определение ИИ

- ***Искусственный интеллект*** — это область науки и инжиниринга, занимающаяся созданием машин и компьютерных программ, обладающих интеллектом. Это название всей области, как физика или математика.
- **Искусственный интеллект (ИИ)** — область научного знания, объединяющая большое число направлений, занимающихся исследованием принципов и закономерностей мыслительной деятельности и моделированием задач, которые традиционно относят к интеллектуальным.
- И т.д.

Рождение ИИ



Trenchard More, John McCarthy, Marvin Minsky, Oliver Selfridge, and Ray Solomonoff.

1956 Dartmouth Summer Research Project on Artificial Intelligence

- Рождение названия «Искусственный Интеллект»
- «The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.»



- **1958, H. A. Simon and Allen Newell:** "within ten years a digital computer will be the world's chess champion" and "within ten years a digital computer will discover and prove an important new mathematical theorem."
- **1965, H. A. Simon:** "machines will be capable, within twenty years, of doing any work a man can do."
- **1967, Marvin Minsky:** "Within a generation ... the problem of creating 'artificial intelligence' will substantially be solved."
- **1970, Marvin Minsky:** "In from three to eight years we will have a machine with the general intelligence of an average human being."

Основные области исследований и разработок в ИИ

- Алгоритмы решения сложных (интеллектуальных) задач (1950-е годы)
- Представление и обработка формализованных знаний (1960-1970 г). Сейчас – онтологии и распределенный интеллект в Интернет
- Искусственные нейронные сети (с 1943г.)
- Техническое зрение и распознавание образов (с 1956г.)
- Промышленные роботы и заводы-автоматы (с 1960г. и с 1980г. соответственно)
- Artificial General Intelligence (AGI) – универсальный ИИ или сильный ИИ (с 2000г.)
- Интеллектуальные и когнитивные роботы (с 2000г.)
- Киборги, нейропротезирование (с 2000г.)
- Интеллектуальное окружение, интернет вещей (с 2000г.)
- Нейроморфные технологии (с 2007г.)

Два типа обучения

- *Обучение по прецедентам (на примерах), или индуктивное обучение*, основано на выявлении эмпирических закономерностей (зависимостей, ассоциаций) в данных.
- *Дедуктивное обучение* предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.
 - По большому счету, не является обучением, его можно считать разновидностью программирования (не алгоритмического)
 - Дедуктивное обучение используется в области экспертных систем, поэтому термины *машинное обучение* и *обучение по прецедентам (на примерах)* можно считать синонимами.

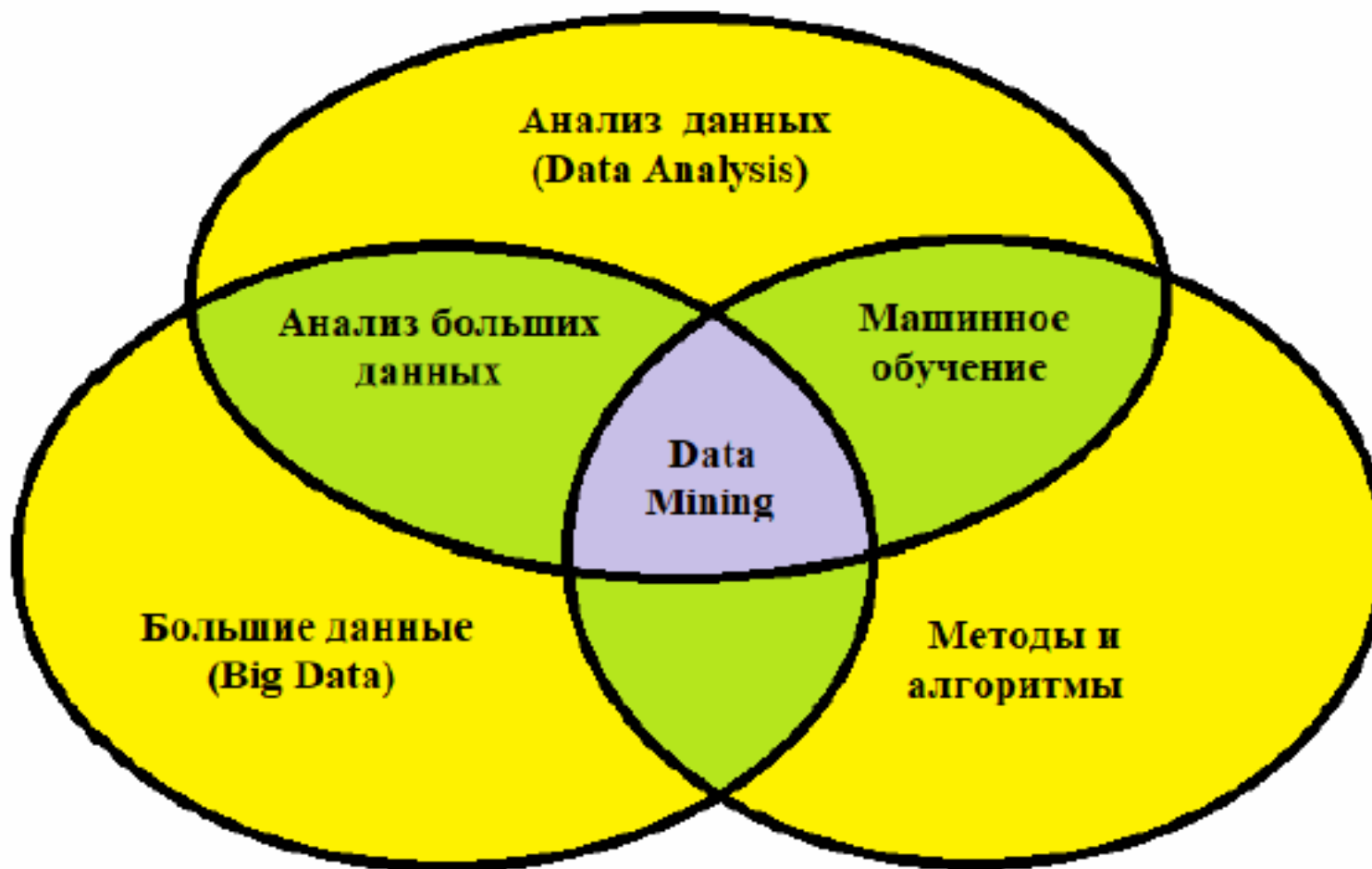
Искусственный интеллект разделяют на два основных направления:

- Символьное (логическое, семиотическое, нисходящее) основано на моделировании высокоуровневых процессов мышления человека, на представлении и использовании формализованных знаний - классический ИИ;
- Нейрокибернетическое (нейросетевое, бионическое, восходящее) основано на моделировании структур мозга, состоящих из нейронов (нейронных сетей) – нейроинформатика
- Есть еще:
 - Гибридное направление в ИИ (гибридные интеллектуальные системы),
 - Когнитивные системы (обучающиеся понятиям),
 - Коллективный интеллект (swarm intelligence),
 - Генетические алгоритмы (эволюционное моделирование),
 - Распознавание образов,
 - Иммунные системы и т.д.

Определения нейронных сетей

- **Нейронные сети** — вид машинного обучения. Это математическая модель, а также её программное или аппаратное воплощение, построенная исходя из попытки моделирования организации и функционирования биологических нейронных сетей.

Место машинного обучения в науке о данных (Data Science)



Пространство признаков

- ***Пространство признаков – это N -мерное пространство, где N – число измеряемых характеристик объектов (признаков, features), выделенное для конкретной задачи. При этом объекты в пространстве признаков задаются N -мерными векторами, каждая компонента которых представляет собой значение определенной характеристики.***

Соотношение между входными параметрами X и результатом Y можно представить простой формулой

The diagram illustrates the components of the regression equation $Y = f(X) + e$. At the top left, the word 'Цель' (Target) has a curved arrow pointing down to the 'Y' in the equation. At the top right, 'Информативные признаки $X_1 \dots X_n$ ' (Informative features) has a curved arrow pointing down to the 'X' in the equation. At the bottom left, 'Модель или сигнал' (Model or signal) has a curved arrow pointing up to the 'f(X)' part of the equation. At the bottom right, 'Ошибка или шум' (Error or noise) has a curved arrow pointing up to the '+ e' part of the equation.

$$Y = f(X) + e$$

В этом уравнении f представляет собой неизвестную функцию, которая связывает входные переменные с целевой переменной Y .

Цель ML-моделирования - точно оценить f , используя данные (признаки).

Источники шума

(на примере автомобилей, датасет – Auto MPG)

- ❑ несовершенные результаты измерений расхода MPG каждой машины, обусловленные небольшими неточностями в измерительных приборах, — *помехи при измерениях* (measurement noise);
- ❑ флуктуации производственного процесса, приводящие к тому, что каждый выпущенный автомобиль будет иметь немого различающиеся показатели MPG, — *шум от производственного процесса* (manufacturing process noise);
- ❑ шум при измерениях таких входных признаков, как вес и мощность в лошадиных силах;
- ❑ отсутствие доступа к более широкому набору признаков, который позволил бы точно определить MPG.

MPG -miles per gallon

Этапы машинного обучения



Задачи машинного обучения



Еще одна классификация МО



Основные направления машинного обучения

- Классическое обучение (простые данные, понятные признаки).
- Обучение с подкреплением (данных нет, но есть среда, с которой можно взаимодействовать).
- Ансамбли (сложные данные).
- Нейросети и глубокое обучение (сложные данные, размытые признаки).

Процесс решения задачи с помощью машинного обучения

1. Определиться со способом решения вашей задачи средствами ML.
2. Выбрать метрику качества.
3. Получить данные и выделить признаки.
4. Определиться с классами моделей.
5. Подготовить данные для обучения и оценки.
6. Провести обучение моделей, оценить результат, повторить шаги, если требуется.

Классическое обучение

- Алгоритмы пришли из математической статистики еще в 1950- годы
- До сих пор широко применяется
- Крупные компании предпочитают нейросети

Классическое обучение.

Виды (то же и для НС):

- Обучение с учителем (данные заранее категоризированы или численные) (supervised learning):
 - классификация (предсказывает категорию);
 - регрессия (предсказывает значения).
- Обучение без учителя (данные никак не размечены) (unsupervised learning):
 - кластеризация (разделение по схожести);
 - уменьшение размерности (обобщение) (найти зависимости);
 - ассоциация (выявить последовательность).
- Обучение с частичным привлечением учителя (данные частично размечены) (semisupervised learning)
- Обучение с подкреплением (данные и поощрение/наказание поступают из внешней среды в процессе решения задачи) (reinforcement learning)

Классификация

- Классификация объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.
- Другими словами, это присвоение объекту какого-либо заранее известного признака. Одежду по цветам, документы по языкам, книги по жанрам, животных по видам или породам.

Использование классификации

- Для спам-фильтров (определение нежелательной рассылки);
- определение языка (сопоставление слов словарям);
- поиск похожих документов (по ключевым словам/фразам);
- анализ тональности (sentiment analyzing) (соотнесение с образцами);
- распознавание рукописных букв и цифр (сопоставление с эталонами);
- определение подозрительных транзакций (выявление аномалий);
- Распознавание лиц, эмоций;
- Распознавание активности человека/толпы
- Распознавание аномалий в визуальной сцене и т.д.

Популярные алгоритмы классификации

- наивный метод Байеса;
- деревья решений;
- логистическая регрессия;
- К-ближайших соседей;
- машины опорных векторов;
- Нейронные сети и т.д.

Наивный Байесовский классификатор

- **Наивный байесовский классификатор** — простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости параметров. (из Википедии)

Пусть у нас есть строка текста O . Кроме того, имеются классы C , к одному из которых мы должны отнести строку. Нам необходимо найти такой класс c , при котором его вероятность для данной строки была бы максимальна. Математически это записывается так:

$$c = \arg \max_C P(C|O)$$

Вычислить $P(C|O)$ сложно. Но можно воспользоваться теоремой Байеса и перейти к косвенным вероятностям:

$$P(C|O) = \frac{P(O|C)P(C)}{P(O)}$$

Так как мы ищем максимум от функции, то знаменатель нас не интересует (он в данном случае константа). Кроме того, нужно взглянуть на строку O . Обычно, нет смысла работать со всей строкой. Намного эффективней выделить из нее определенные признаки (features). Таким образом формула примет вид:

$$P(C|o_1 o_2 \dots o_n) = \frac{P(o_1 o_2 \dots o_n | C) P(C)}{P(o_1 o_2 \dots o_n)}$$

Знаменатель нас не интересует.

так как знаменатель не зависит от C и значения параметров заданы, так что знаменатель — константа.

Здесь включаем «наивное» предположение о том, что переменные O зависят только от класса C , и не зависят друг от друга. Это сильно упрощение, но зачастую это работает. Числитель примет вид:

$$P(C)P(o_1|C)P(o_2|Co_1)...P(o_n|Co_1o_2...o_{n-1}) = P(C)P(o_1|C)P(o_2|C)..P(o_n|C) = P(C) \prod_i (o_i|C)$$

Финальная формула примет вид:

$$c = \arg \max_{c \in C} P(c|o_1o_2...o_n) = \arg \max_{c \in C} P(c) \prod P(o_i|c) \quad (1)$$

Т.е. все что нужно сделать, это вычислить вероятности $P(C)$ и $P(O|C)$. Вычисление этих параметров и называется тренировкой классификатора.

Деревья решений

- Машина бинарно разделяет все данные по вопросам, ответы на которые «да» или «нет». Пусть даже эти вопросы и не всегда адекватны с точки зрения обычного человека, например, «зарплата заёмщика больше, чем 12345 рублей?», но они придуманы так, чтобы на каждом вопросе разбиение было максимально точным.

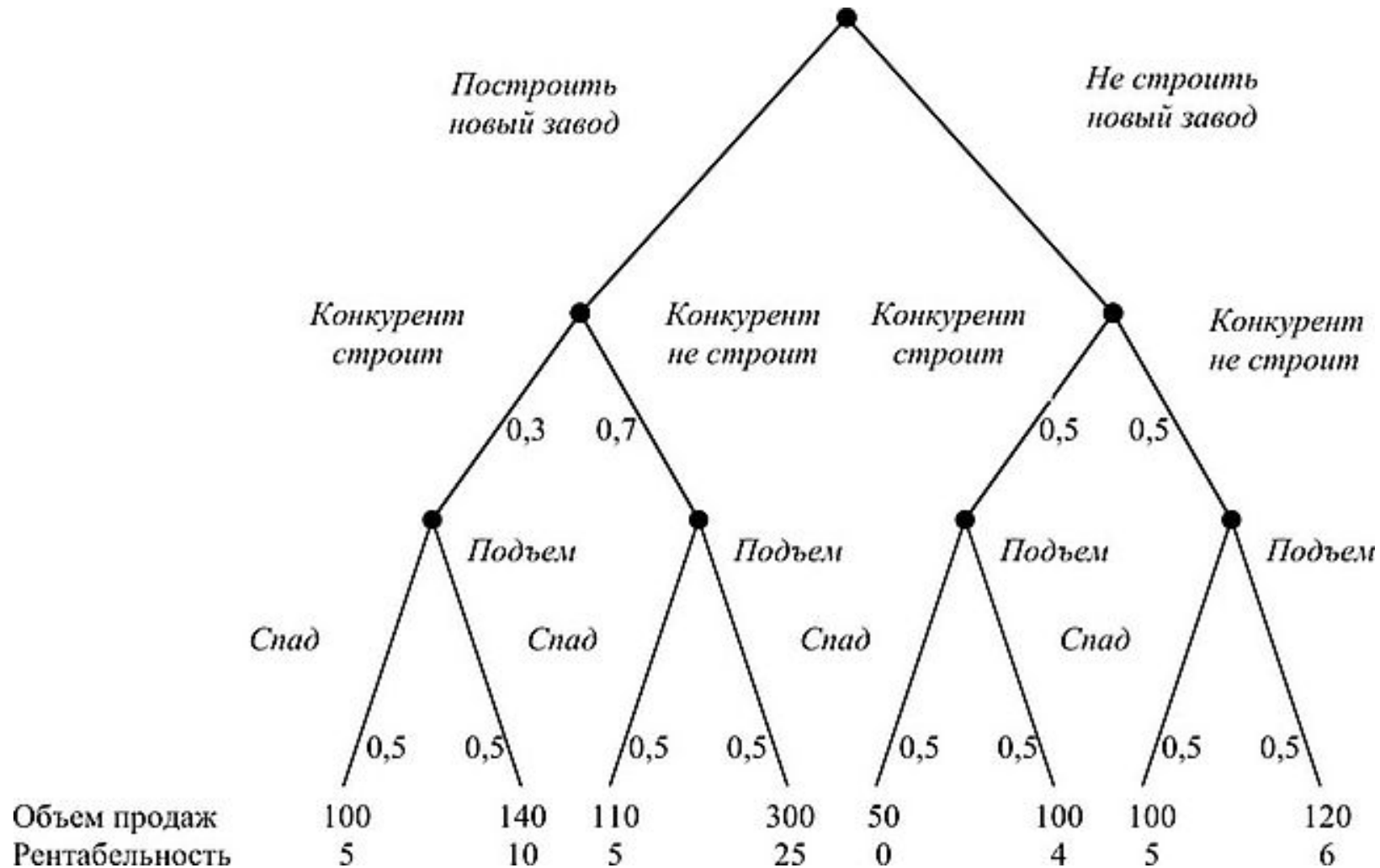
Структура дерева решений

- **Корень (root)** — самый верхний узел. С него начинается разбиение.
- **Внутренние узлы** — узлы, в которых происходит проверка по какому-либо признаку и выбор ветви.
- **Ветви (branches)** — переходы между узлами, соответствующие условию (например, возраст < 30).
- **Листья (leaves)** — конечные узлы. Именно в них алгоритм даёт предсказание: класс (для классификации) или числовое значение (для регрессии).

Виды деревьев решений

- **Дерево классификации** — предсказывает категориальную переменную (например, одобрить/отклонить кредит).
- **Дерево регрессии** — предсказывает непрерывную величину (например, цену квартиры).

Пример 1 дерева решений



Пример 2 дерева решений



Общий алгоритм построения дерева решений

- Берём всю обучающую выборку в корневом узле.
- Для каждого признака и возможного порога ищем лучшее разбиение по выбранному критерию.
- Разбиваем узел на дочерние по найденному лучшему условию.
- Повторяем для каждого дочернего узла, пока не выполнится критерий остановки.
- В листьях присваиваем предсказание (чаще всего — мажоритарный класс или среднее значение)

Основные алгоритмы деревьев решений

ID3 (Quinlan, 1986)

Первый вариант. Использует энтропию и Information Gain (метрика, которая показывает, насколько снижается неопределённость-энтропия после разбиения данных по конкретному признаку). Работает только с категориальными признаками.

C4.5 (Quinlan, 1993)

Улучшение ID3: поддержка числовых признаков, обработка пропусков, обрезка дерева (pruning).

CART (Classification and Regression Trees, Breiman, 2001)

Самый распространённый сегодня. Всегда строит **бинарные** разбиения, использует Gini для классификации и MSE для регрессии. Именно он реализован в sklearn.

Алгоритм ID3

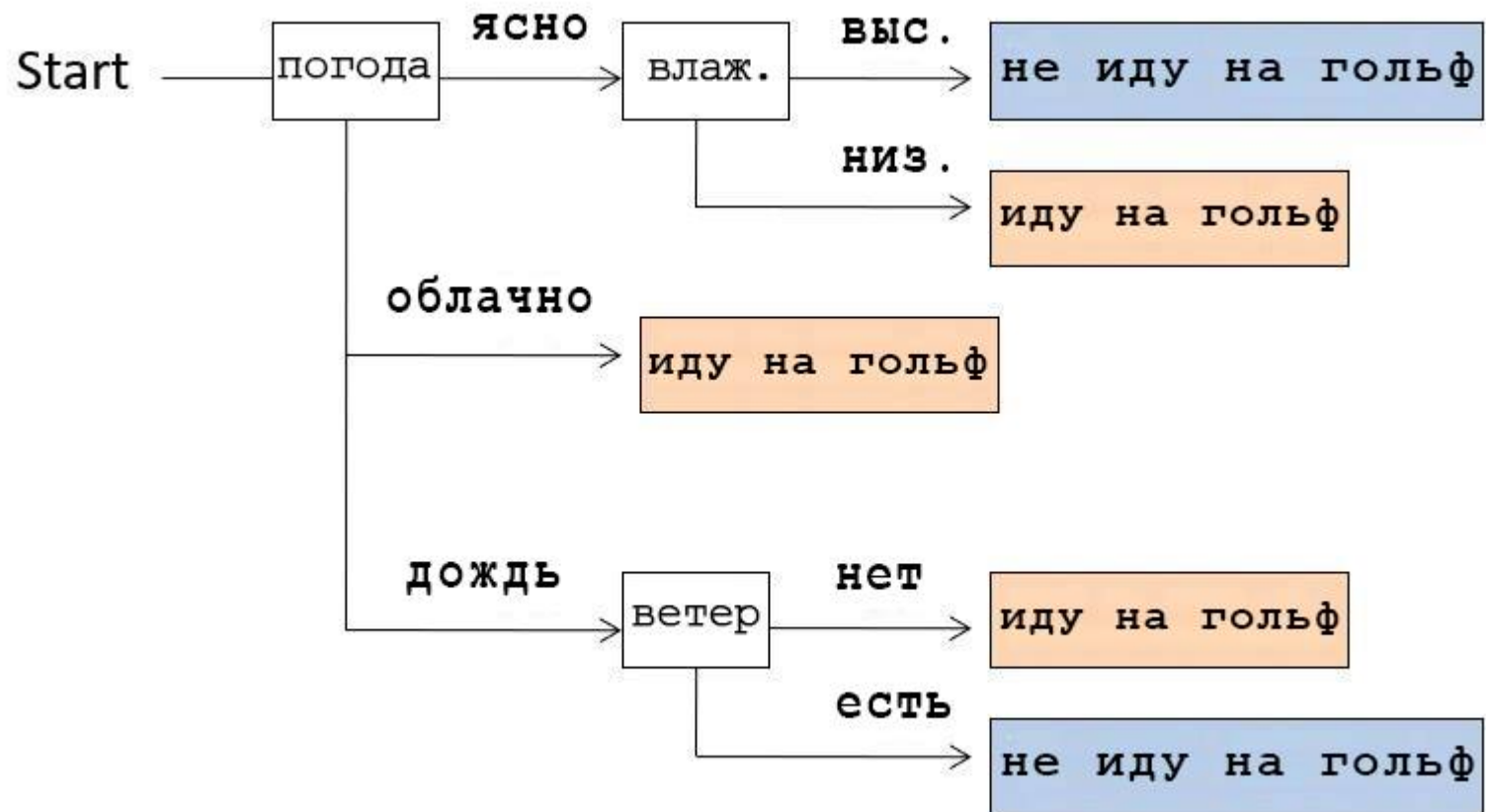
ID3(Таблица примеров, Целевой признак, Признаки)

1. Если все примеры положительны, то вернуть узел с меткой «+».
2. Если все примеры отрицательны, то вернуть узел с меткой «-».
3. Если множество признаков пустое, то вернуть узел с меткой, которая больше других встречается в значениях целевого признака в примерах.
4. Иначе:
 1. A — признак, который лучше всего классифицирует примеры (с максимальной информационной выгодой).
 2. Создать корень дерева решения; признаком в корне будет являться A.
 3. Для каждого возможного значения $A(v_i)$:
 1. Добавить новую ветвь дерева ниже корня с узлом со значением $A=v_i$.
 2. Выделить подмножество $Examples(v_i)$ примеров, у которых $A=v_i$.
 3. Если подмножество примеров пусто, то ниже этой новой ветви добавить узел с меткой, которая больше других встречается в значениях целевого признака в примерах.
 4. Иначе, ниже этой новой ветви добавить поддереву, вызывая рекурсивно ID3($Examples(v_i)$, Целевой признак, Признаки)
5. Вернуть корень.

Например, у нас есть следующий набор данных (дата сет): погода, температура, влажность, ветер, игра в гольф. В зависимости от погоды и остального, мы ходили (○) или не ходили (×) играть в гольф. Предположим, что у нас есть 14 сложившихся вариантов.

№	погода	темп.	влажн.	ветер	гольф
1	ясно	жарко	выс.	нет	×
2	ясно	жарко	выс.	есть	×
3	облачно	жарко	выс.	нет	○
4	дождь	тепло	выс.	нет	○
5	дождь	холодно	норм.	нет	○
6	дождь	холодно	норм.	есть	×
7	облачно	холодно	норм.	есть	○
8	ясно	тепло	выс.	нет	×
9	ясно	холодно	норм.	нет	○
10	дождь	тепло	норм.	нет	○
11	ясно	тепло	норм.	есть	○
12	облачно	тепло	выс.	есть	○
13	облачно	жарко	норм.	нет	○
14	дождь	тепло	выс.	есть	×

Полученное дерево решений



```

import math
import pandas as pd

from functools import reduce
# Дата сет
d = {
    "Погода":["ясно","ясно","облачно","дождь",
    ", "дождь", "дождь", "облачно", "ясно", "ясно",
    ", "дождь", ""],
    "Температура":["Жарко", "Жарко", "Жарко", "Тепло", "Холодно", "Холодно",
    "Холодно", "Тепло", "Холодно", ""],
    "Влажность":["Высокая", "Высокая", "Высокая", "Высокая", "Норм", "Норм", "Норм", "Норм", "Норм", ""],
    "Ветер":["Нет", "Есть", "Нет", "Нет", "Нет", "Есть", "Есть", "Есть", "Нет", "Нет", "Нет", "Есть", "Есть", "Нет", ""],
    "# Последний массив - это наша целевая переменная, показывающая результат, основывающийся на предыдущих данных.
    "Гольф":["x", "x", "o", "o", "o", "x", "o", "x", "o", "o", "o", "o", "o", "x"],
}

```

```

df0 = pd.DataFrame(d)
# Лямбда-выражение для распределения значений, аргумент - pandas.Series,
# возвращаемое значение - массив с количеством каждого из значений
# Из вводных данных s с помощью value_counts() находим частоту каждого из значений,
# и пока в нашем словаре есть элементы, будет работать цикл, запускаемый items().
# Чтобы выходные данные не менялись с каждым запуском цикла, мы используем sorted,
# который меняет порядок от большего к меньшему
# В итоге, генерируется массив, содержащий строку из следующих компонентов: ключ (k) и значение cstr =
lambda s:[k+":"+str(v) for k,v in sorted(s.value_counts().items())]

```

```

# Структура данных Decision Tree
tree = {
# name: Название этого нода (узла)
"name":"decision tree "+df0.columns[-1]+"
"+str(cstr(df0.iloc[:,-1])),
# df: Данные, связанные с этим нодом
(узлом)
"df":df0,
# edges: Список ребер (ветвей),
выходящих из этого узла,
# или пустой массив, если ниже нет
листового узла.
"edges":[],
}
# Генерацию дерева, у узлов которого
могут быть ветви, сохраняем в open
open = [tree]
# Лямба-выражение для вычисления
энтропии.
# Аргумент - pandas.Series,
возвращаемое значение - число
энтропии

```

```

entropy = lambda s:-reduce(lambda
x,y:x+y,map(lambda
x:(x/len(s))*math.log2(x/len(s)),s.value_cou
nts()))
# Зацикливаем, пока open не станет
пустым
while(len(open)!=0):
# Вытаскиваем из массива open первый
элемент,
# и вытаскиваем данные, хранящиеся в
этом узле
n = open.pop(0)
df_n = n["df"]
# В случае, если энтропия этого узла
равна 0, мы больше не можем
вырастить из него новые ветви
# поэтому прекращаем ветвление от
этого узла
if 0==entropy(df_n.iloc[:,-1]):
continue

```

```

# Создаем переменную, в которую
будем сохранять список значений
атрибута с возможностью разветвления
attrs = {}
# Исследуем все атрибуты, кроме
последнего столбца класса атрибутов
for attr in df_n.columns[:-1]:
# Создаем переменную, которая хранит
значение энтропии при ветвлении с
этим атрибутом,
# данные после разветвления и
значение атрибута, который
разветвляется.
attrs[attr] = {"entropy":0,"dfs":[],"values":[]}
# Исследуем все возможные значения
этого атрибута.
# Кроме того, sorted предназначен для
предотвращения изменения порядка
массива,
# из которого были удалены
повторяющиеся значения атрибутов,
при каждом его выполнении.
for value in sorted(set(df_n[attr])):

```

```

# Фильтруем данные по значению
атрибута
df_m = df_n.query(attr+"=="+"value+""")
# Вычисляем энтропию, данные и
значения сохраняем
attrs[attr]["entropy"] += entropy(df_m.iloc[:,-
1])*df_m.shape[0]/df_n.shape[0]
attrs[attr]["dfs"] += [df_m]
attrs[attr]["values"] += [value]
pass
pass
# Если не осталось ни одного атрибута,
значение которого можно разделить,
# прерываем исследование этого узла.
if len(attrs)==0:
continue
# Получаем атрибут с наименьшим
значением энтропии
attr = min(attrs,key=lambda
x:attrs[x]["entropy"])
# Добавляем каждое значение
разветвленного атрибута

```

```

# и данные, полученные после
разветвления, в наше дерево и в open.
for d,v in
zip(attrs[attr]["dfs"],attrs[attr]["values"]):
m =
{"name":attr+"="+v,"edges":[],"df":d.drop(co
lums=attr)}
n["edges"].append(m)
open.append(m)
pass
# Выводим дата сет
print(df0,"\n-----")
# Метод преобразования дерева в
символы, аргументы - tree:структура
данных древа,
# indent:присоединяемый к дочернему
узлу indent,
# Возвращаемое значение - символьное
представление древа.
# Этот метод вызывается рекурсивно
для преобразования всех данных в
дереве в символы.
def tstr(tree,indent=""):

```

```

# Создаем символьное представление
этого узла.
# Если этот узел является листовым
узлом (количество элементов в массиве
ребер равно 0),
# частотное распределение последнего
столбца данных df, связанных с
деревом, преобразуется s =
indent+tree["name"]+str(cstr(tree["df"].iloc[:,
-1]) if len(tree["edges"])==0 else "")+"\n#
Зацикливаем все ветви этого узла.
for e in tree["edges"]:
# Добавляем символьное
представление дочернего узла к
символьному представлению
родительского # Добавляем еще
больше символов к indent этого узла.
s += tstr(e,indent+" ")
pass
return s
# Выводим древо в его символьном
представлении.
print(tstr(tree))

```

Если запустить вышеописанную программу, наш Decision Tree будет представлен в виде символьной таблицы, как показано ниже.

```
decision tree Гольф ['x:5', 'o:9']
```

```
  Погода=Ясно
```

```
    Влажность=Обычная['o:2']
```

```
    Влажность=Высокая['x:3']
```

```
  Погода=Облачно['o:4']
```

```
  Погода=Дождь
```

```
    Ветер=Есть['x:2']
```

```
    Ветер=Нет['o:3']
```

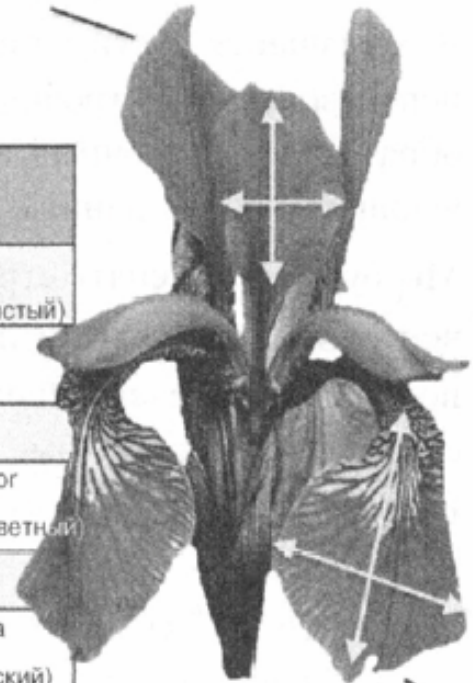
Пример: классификация ирисов

Выборки
(образцы, наблюдения)

	Длина чаше- листика	Ширина чаше- листика	Длина лепестка	Ширина лепестка	Метка класса
1	5.1	3.5	1.4	0.2	Setosa (ирис щетинистый)
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor (ирис разноцветный)
...					
150	5.9	3.0	5.0	1.8	Virginica (ирис виргинский)

Признаки
(атрибуты, измерения, размерности)

Метки классов
(цели)



Пример с использованием Scikit-learn (классификация ирисов)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(max_depth=3, criterion='gini',
    random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Визуализация

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
plot_tree(clf, feature_names=['sepal
length','sepal width','petal length','petal
width'],
class_names=['setosa','versicolor','virginic
a'], filled=True)
plt.show()
```

Преимущества деревьев решений

- **Интерпретируемость.** Легко объяснить, почему модель сделала именно такое предсказание.
- **Наглядность.** Дерево можно полностью визуализировать.
- **Не требует нормализации.** Масштаб признаков не влияет на разбиения.
- **Работает с числовыми и категориальными признаками.**
- **Мало предположений** о распределении данных.
- **Быстрое предсказание** после обучения.

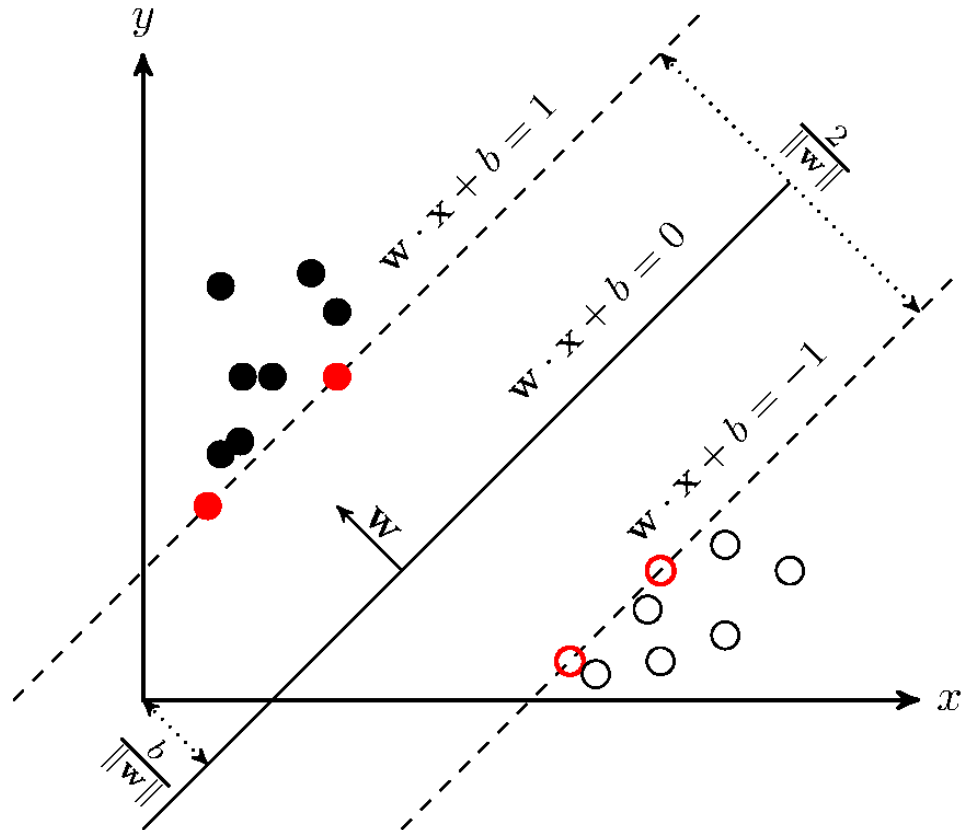
Недостатки

- **Склонно к переобучению** без ограничений.
- **Нестабильность.** Небольшое изменение в данных может сильно изменить структуру дерева.
- **«Жадный» поиск.** На каждом шаге выбирает локально лучшее разбиение, а не глобально оптимальное.
- **Ориентированность по осям.** Разбиения всегда параллельны осям признаков.
- **Смещение в пользу признаков с большим числом значений** (особенно у старых алгоритмов).

Метод опорных векторов (Support Vector Machine)

- Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с наибольшим зазором в этом пространстве.
- Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. *Разделяющей гиперплоскостью* будет гиперплоскость, создающая наибольшее расстояние до двух параллельных гиперплоскостей. Алгоритм основан на допущении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.
- Логика SVM довольно проста (применительно к двум классам) — он пытается найти способ, как провести две прямые (в пространстве признаков) между определенными категориями, чтобы между ними образовался наибольший зазор.

Метод опорных векторов (2)



Если данные **не являются линейно разделимыми**, SVM позволяют проецировать их в пространство **большей размерности** с помощью функции ядра.

Пример:

- Точки на плоскости, разделённые окружностью.
- Можно добавить третью координату $z = x^2 + y^2$. Тогда данные станут линейно разделимыми в 3D.

Ядро (kernel) — это функция ($K(x_i, x_j)$), которая **вычисляет скалярное произведение** в новом пространстве, **не выполняя само проецирование**.

Наиболее распространённые ядра:

- Полиномиальное (однородное): $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d$
- Полиномиальное (неоднородное): $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$
- Радиальная базисная функция: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$, для $\gamma > 0$
- Радиальная базисная функция Гаусса: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$
- Сигмоид: $k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \mathbf{x} \cdot \mathbf{x}' + c)$, для почти всех $\kappa > 0$ и $c < 0$

Пример с scikit-learn

```
from sklearn import svm
from sklearn.datasets import make_blobs
# создаём данные
X, y = make_blobs(n_samples=50, centers=2,
                  random_state=6)
# создаём классификатор SVM
clf = svm.SVC(kernel='linear', C=1.0)
clf.fit(X, y)
# предсказание
new_point = [[2, 0.5]]
prediction = clf.predict(new_point)
print("Класс:", prediction[0])
```

Достоинства

- Эффективно для наборов данных с несколькими характеристиками, например финансовых или медицинских данных.
- Эффективно в случаях, когда количество признаков превышает количество точек данных.
- Использует подмножество обучающих точек в функции принятия решения, называемое векторами поддержки, что позволяет экономить память.
- Для функции принятия решения можно указать различные функции ядра. Вы можете использовать стандартные ядра, но также можно указать собственные.

Недостатки

- Если количество признаков намного превышает количество точек данных, крайне важно избегать переобучения при выборе ядерных функций и регуляризации.
- Метод опорных векторов не дает прямых оценок вероятности. Они рассчитываются с помощью дорогостоящей пятикратной перекрестной проверки.
- Лучше всего работает на небольших наборах данных из-за длительного времени обучения.

Регрессия

- Регрессия – это набор статистических методов исследования влияния одной или нескольких независимых переменных на зависимую переменную.
- Сегодня используют для:
 - прогноз стоимости ценных бумаг (экстраполяция)
 - анализ спроса, объема продаж (выявления зависимостей);
 - медицинские диагнозы (прогнозы лечения);
 - любые зависимости числа от времени (временные ряды).
- Популярные алгоритмы:
 - Линейная или Полиномиальная Регрессия.
- В отличие от классификации на выходе имеем не номер класса, а предсказываемое число. Можно считать, что в этом случае машина учится вычислению произвольной неизвестной функции, заданную обучающими примерами.

Обучение без учителя

- **Обучение без учителя** (Unsupervised Learning) это более позднее изобретение, но все же 1970-х, и в практических задачах используется редко.
- **Обучение без учителя** (самообучение, спонтанное обучение, *Unsupervised learning*) — один из способов машинного обучения, при котором испытываемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора. С точки зрения кибернетики, это является одним из видов кибернетического эксперимента. Как правило, это пригодно только для задач, в которых известны описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами. (из Википедии)

Решаемые с помощью обучения без учителя задачи

- Задача кластеризации.
- Поиск ассоциаций.
- Сокращение размерности данных.
- Визуализация данных.

В определенной степени каждая из последних трех задач является производной от первой или ее частным случаем.

Кластеризация

- ***Кластеризация*** (или кластерный анализ) — это задача разбиения множества объектов на группы, называемые кластерами. При этом признак (по которому происходит группирование в кластеры) заранее неизвестен. Машина сама определяет, как определять кластеры.

Примеры решаемых задач с использованием кластеризации:

- сегментация рынка (типов покупателей, лояльности);
- объединение близких точек на карте (зуммирование);
- сжатие изображений (снижение объема);
- анализ и разметка новых данных (из уже имеющихся);
- детекторы аномального поведения (всплески активности).

Популярные алгоритмы: Метод К-средних, Mean-Shift, DBSCAN

Сложность кластеризации

- Нет однозначного критерия качества кластеризации. Существует ряд эвристических критериев, а также ряд бескритериальных алгоритмов, выполняющих вполне осмысленную кластеризацию, но дающих на одних и тех же данных разные результаты.
- Число кластеров, как правило, заранее неизвестно и задается субъективно. (но в нейронных сетях есть растущие НС, в частности, модель ART, в которой количество кластеров увеличивается по мере необходимости)
- На результат кластеризации существенное влияние оказывает выбранная метрика расстояния, которая, как правило, также выбирается субъективно.

Несмотря на сложности, кластеризация позволяет:

- Улучшить понимание данных за счет выявления их кластерной структуры: разбиение объемной выборки на группы схожих объектов может упростить дальнейшую обработку данных за счет применения к каждому кластеру своих методов анализа.
- Осуществить сжатие данных. В данном случае подразумевается сокращение объемной выборки за счет работы с наиболее яркими представителями кластеров (групп схожих объектов).
- Выявить новизну в массиве данных: обнаружить нетипичные объекты, которые не удастся отнести ни к одному кластеру.
- Решить задачу таксономии: построить древообразную иерархическую структуру, упорядочивающую исходные данные. Ее построение достигается за счет дробления крупных кластеров на более мелкие, которые в свою очередь также дробятся на еще более мелкие. Визуально таксономия отображается в виде графика – *дендрограммы*.

Снижение размерности (обобщение)

- **Снижение размерности** — это преобразование данных, состоящее в уменьшении числа переменных (признаков) путём получения главных переменных.
- Алгоритм объединяет конкретные признаки в абстракции более высокого уровня.
- **Задача сокращения размерности данных** состоит в следующем. Существует большой (значительно большой) объем признаковых описаний объектов. Причем этот объем обуславливается внушительным количеством измерений признакового пространства. Необходимо представить те же данные в пространстве меньшей размерности, при этом минимизировав потери информации. Группировка по кластерам и является одним из вариантов решения проблемы.

Снижение размерности используется в:

- рекомендательных системах (чаще всего);
- визуализации (для адаптации изображений);
- определении тематики и поиска похожих документов (в многоуровневых классификаторах);
- анализе фейковых изображений (не вписывающихся в общие признаки);
- риск-менеджменте (оценка последствий действий).

Популярные методы снижения размерности

- метод главных компонент (*principal component analysis*, PCA);
- метод независимых компонент (ICA); • Многомерное шкалирование (*Multidimensional scaling*, MDS);
- сингулярное разложение (SVD);
- латентное размещение Дирихле (LDA);
- латентно-семантический анализ (LSA, LSA, GLSA), t-SNE (для визуализации).
- Свертка в сверточных нейронных сетях

Поиск ассоциаций (правил)

- *Метод ассоциаций заключается в поиске закономерностей в наборах объектов.*
- В настоящее время метод ассоциаций используют для:
 - прогноза акций и распродаж (каким образом спроектировать предложение);
 - анализа товаров, покупаемых вместе (для комплексных предложений);
 - расстановки товаров на полках (чтобы связанные товары стояли рядом, или наоборот);
 - анализ паттернов поведения на веб-сайтах (оценка восприятия контента);
- Популярные алгоритмы: Apriori, Euclat, FP-growth.
- При всей его практической ценности, поиск правил — самая слабо проработанная категория среди всех методов обучения.

Обучение с подкреплением

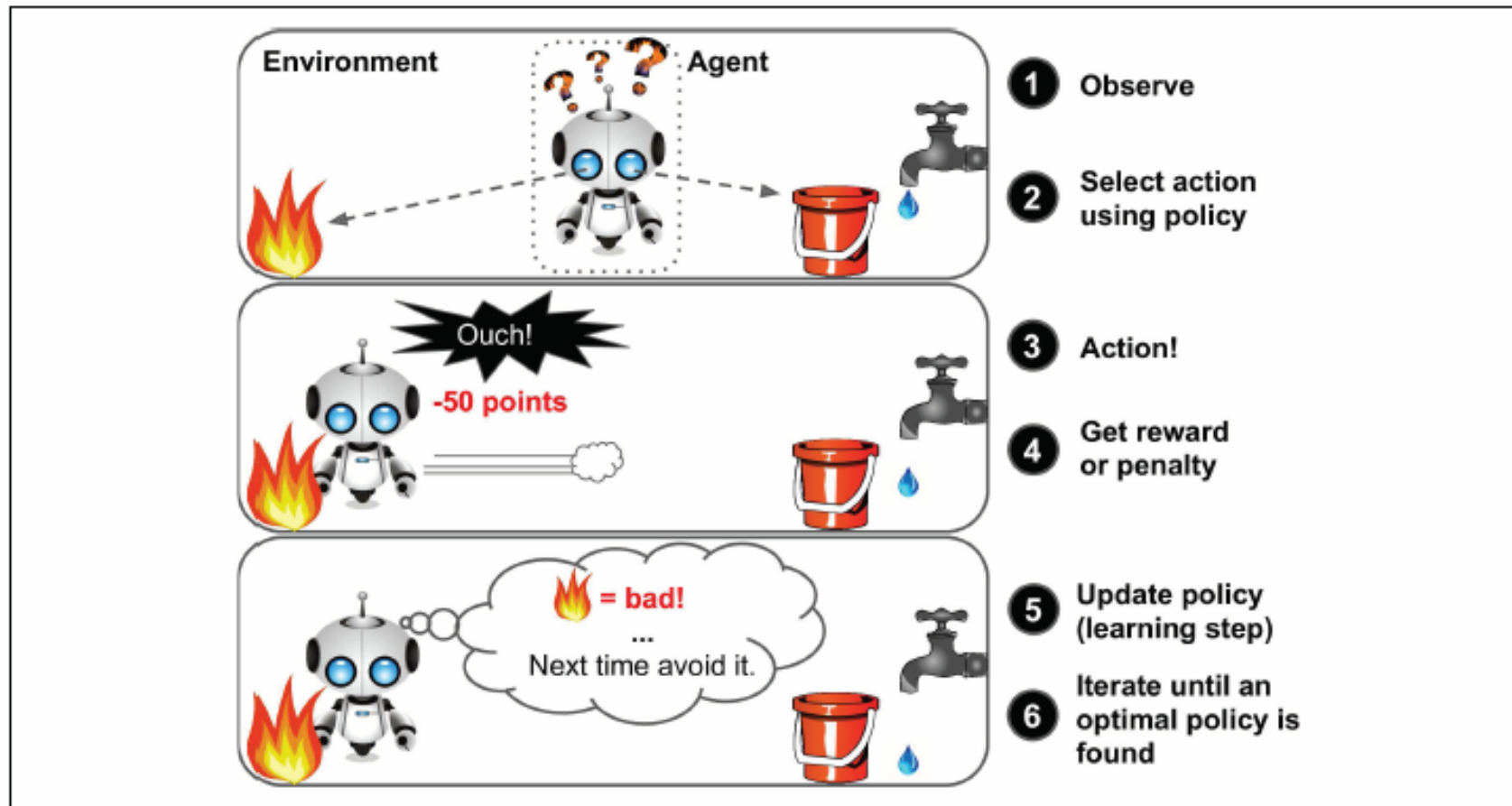
- *Метод машинного обучения, при котором происходит обучение модели, которая не имеет сведений о среде, но имеет возможность производить какие-либо действия в ней.*
- Постановка задачи обучения с подкреплением:
 - Есть среда, в которой действует агент. У среды есть состояние.
 - Агент может совершать действия.
 - Действия приносят некий результат (поощрение-reward или наказание-punishment) (не обязательно мгновенно).
 - Надо научиться такой модели поведения, которая максимизирует reward или/и минимизирует punishment.
-



Обучение с подкреплением (2)

- Наиболее часто используется для:
 - самоуправляемых автомобилей (навигация и предотвращение ДТП);
 - управления роботами;
 - игр (виртуальные персонажи);
 - автоматической торговли (в частности, автотрейдинг ценных бумаг);
 - управления ресурсами предприятий (системы JIT).
- Популярные алгоритмы: Q-Learning, SARSA, DQN, A3C, генетический алгоритм.

Обучение с подкреплением (3)



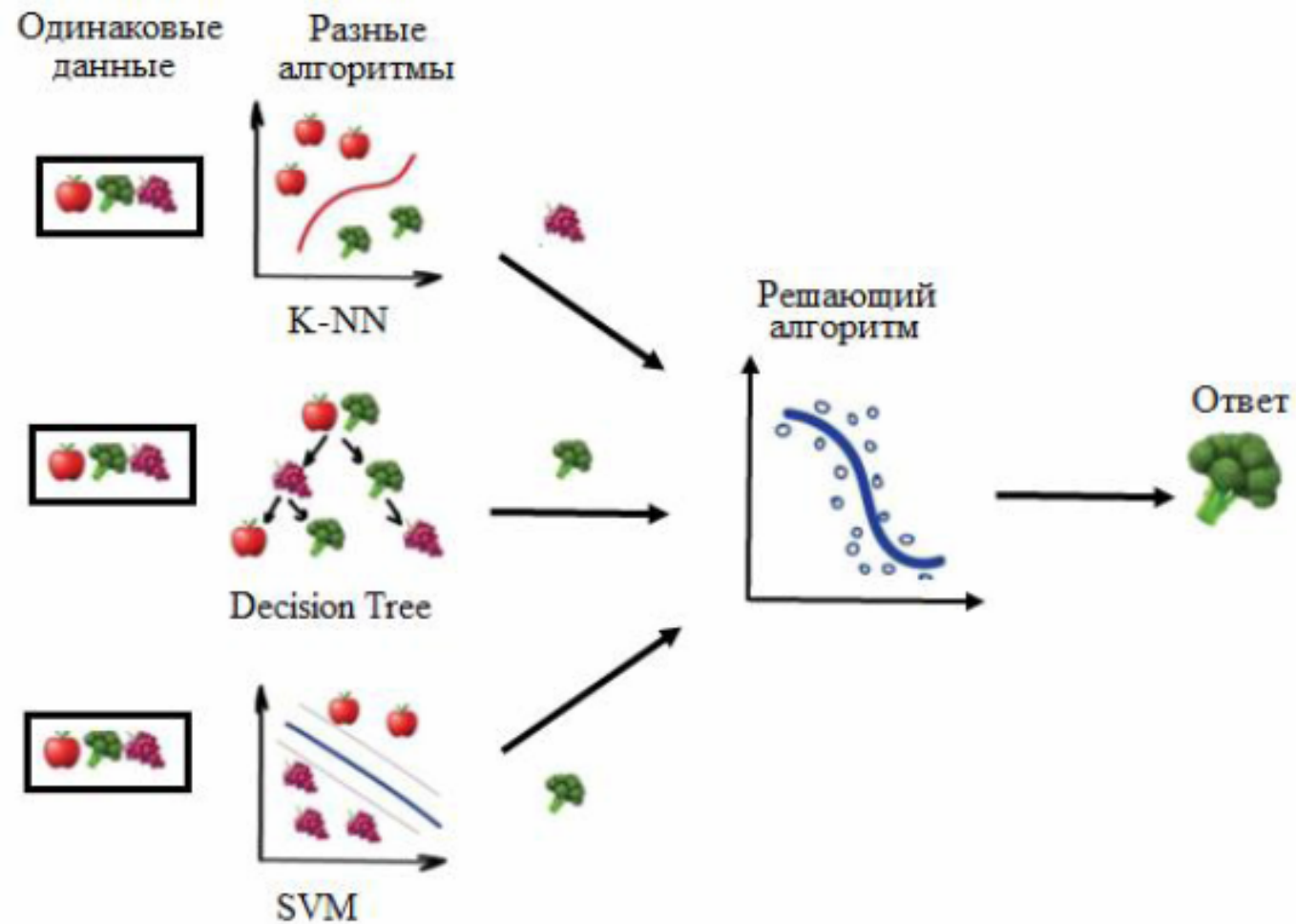
Ансамбли

- *Использование нескольких связанных между собой алгоритмов обучения, которые направлены на повышение точности друг друга.*
- Сегодня ансамбли используют для:
 - всего, где подходят классические алгоритмы (но с многократно большей точностью);
 - поисковые системы (классическая задача для деревьев);
 - компьютерное зрение (определение расстояний, форм и т.д.);
 - распознавание объектов (в том числе и живых).
- Популярные алгоритмы: Random Forest, Gradient Boosting.
- Методы создания ансамбля: стекинг, беггинг и бустинг

Стекинг

- Обучаем несколько разных алгоритмов и передаём их результаты на вход последнему, который принимает итоговое решение.
- Самое главное здесь, чтобы задача решалась разными алгоритмами, ведь учить один и тот же алгоритм на одних и тех же данных не имеет смысла. Какие алгоритмы выбирать в качестве начальных – дело вкуса, но в качестве решающего алгоритма лучше брать регрессию

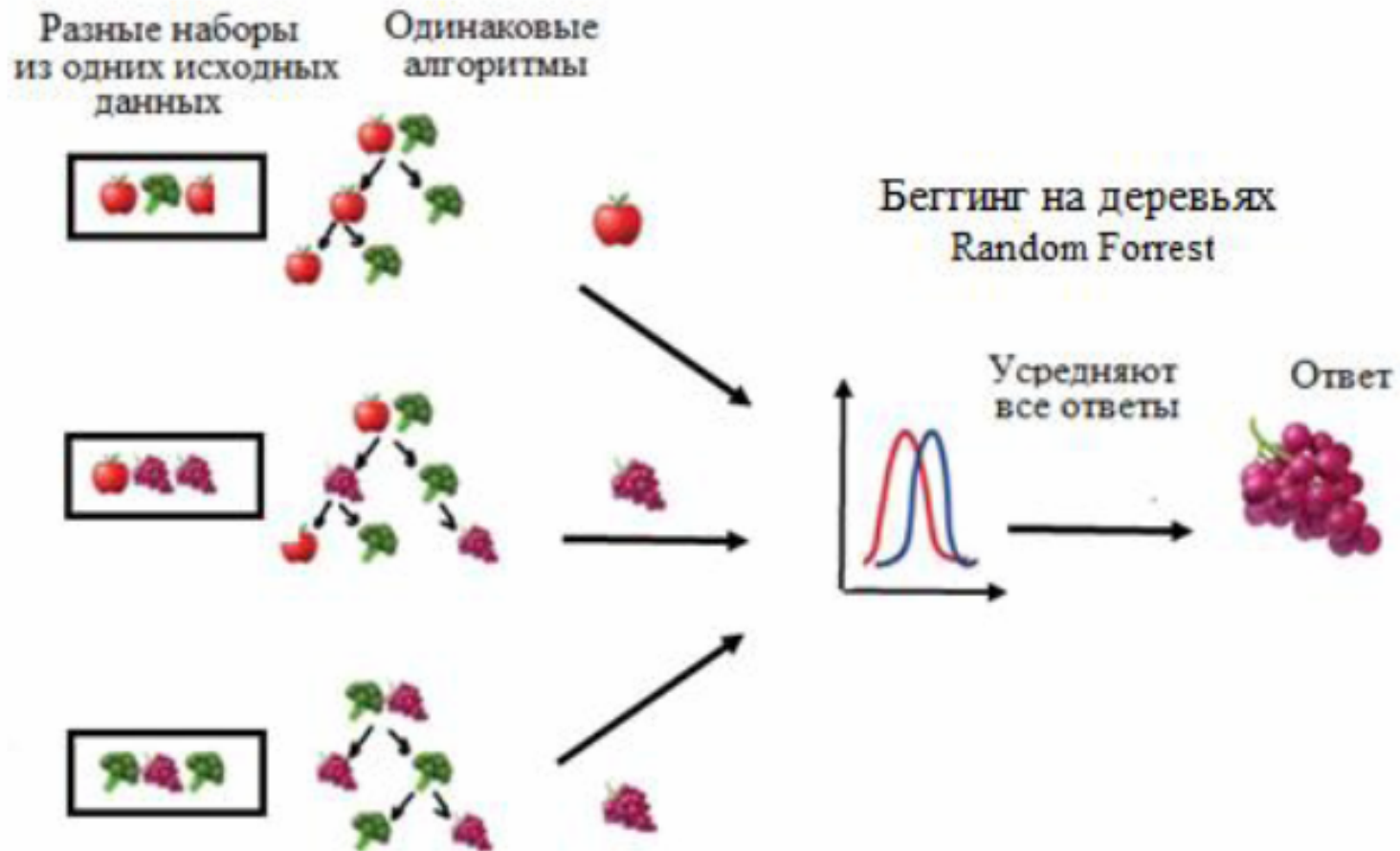
Стекинг (2)



Беггинг (Bootstrap AGGregaratING)

- Сущность беггинга заключается в следующем.
 - Выбираем базовый алгоритм.
 - Обучаем алгоритм на случайных выборках из исходных данных много раз.
 - На финальном этапе результаты просто усредняются.
- Самый популярный пример беггинга — алгоритм Random Forest. Именно он применяется в распознавании образов. Нейросеть слишком медленно работает в реальном времени на обычных мощностях, а беггинг может считать свои деревья параллельно на всех шейдерах видеокарты.
- Легко распараллеливается

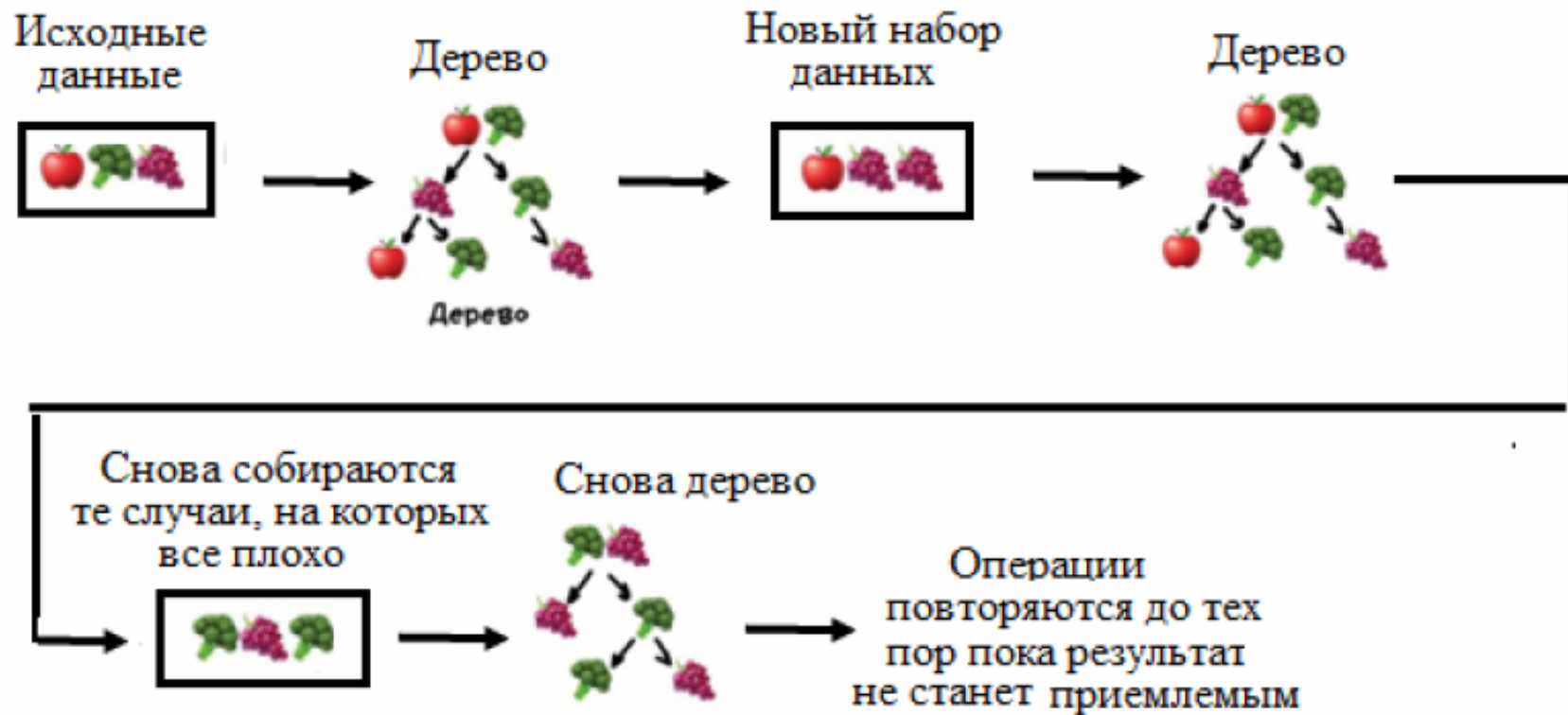
Беггинг (2)



Бустинг

- Алгоритмы обучаются последовательно, при этом каждый следующий уделяет особое внимание ситуациям, в которых ошибся предыдущий.
- Так же, как и в предыдущем случае, делаются выборки из исходных данных, но уже не совсем случайно.
- В каждую следующую выборку обязательно включается часть тех данных, на которых ошибся предыдущий алгоритм.
- Плюсом является высокая точность классификации,
- Минус – последовательная обработка.

Бустинг (2)



Глубокое обучение и нейросети

- **Нейронная сеть** (также *искусственная нейронная сеть, ИНС*) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей.

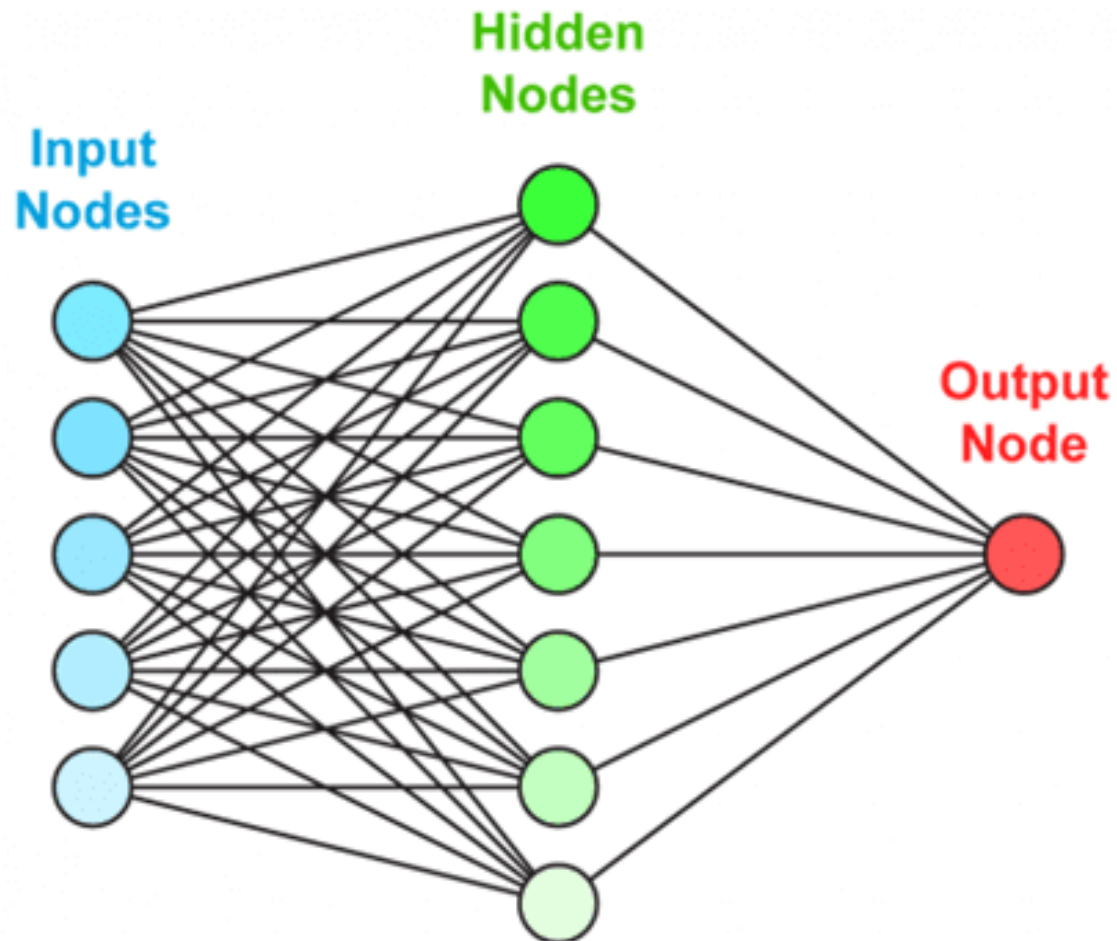
Нейронные сети

- Сегодня используют для:
 - вместо всех вышеперечисленных алгоритмов;
 - распознавание объектов на фото и видео (лица, эмоций, номерных знаков, дефектов в изделиях, препятствий и знаков на дороге и т.д.);
 - распознавание и синтез речи;
 - генерация изображений (формирование стиля, сжатие, расшифровка, генерация правдоподобного персонажа или копии на фото или видео);
 - машинный перевод (обработка текстов).
 - генерация подходящего ответа в диалоговых системах
 - и т.д.

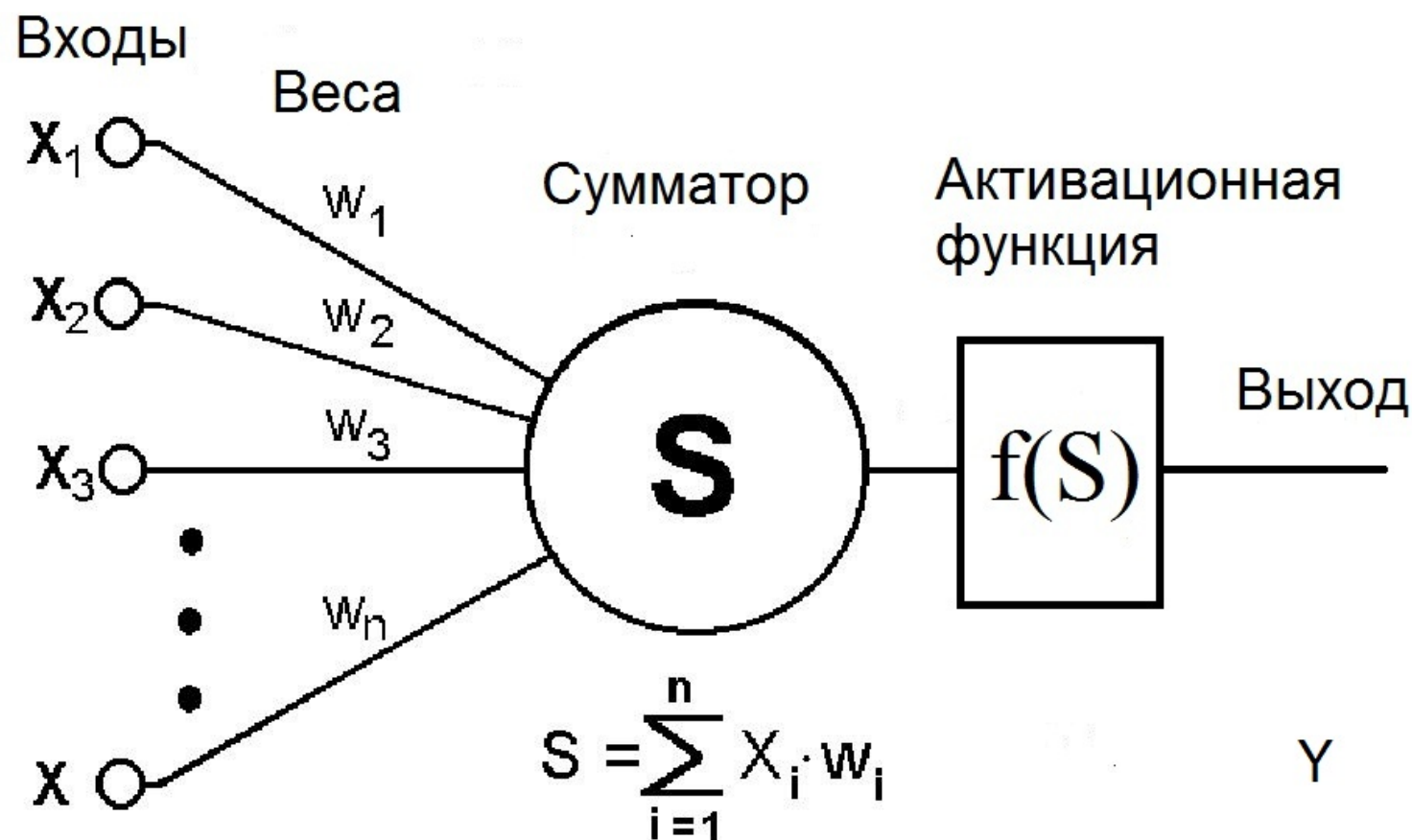
Нейронные сети (2)

- Наиболее популярные архитектуры:
 - Персептрон (MLP);
 - Сверточные сети (CNN);
 - Рекуррентные Сети (RNN, LSTM, GRU);
 - Автоэнкодеры (AE);
 - Генеративно-состязательные сети;
 - Трансформеры (сети с вниманием).

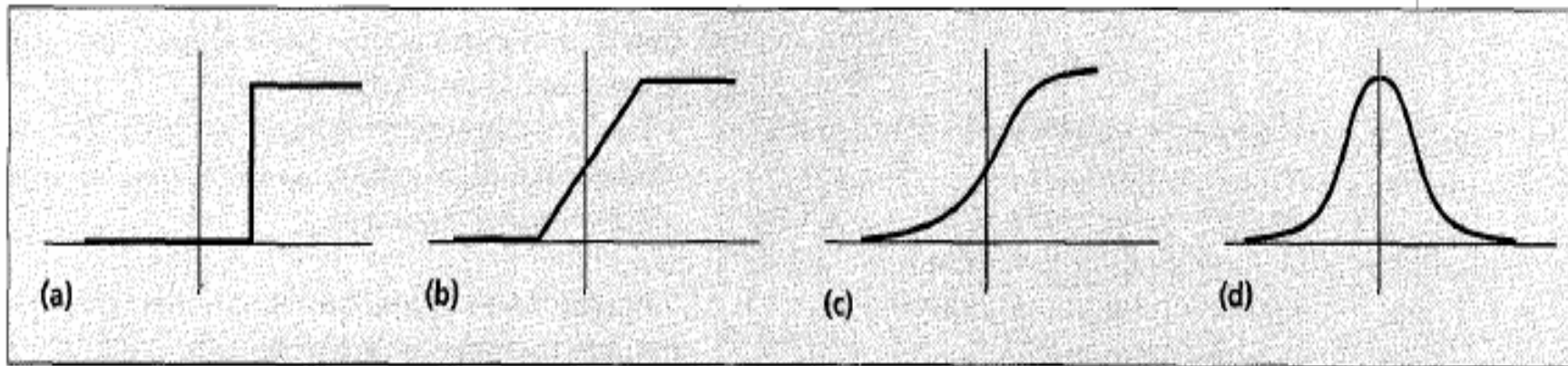
Типичная нейронная сеть



Типичная модель нейрона



Виды активационной функции

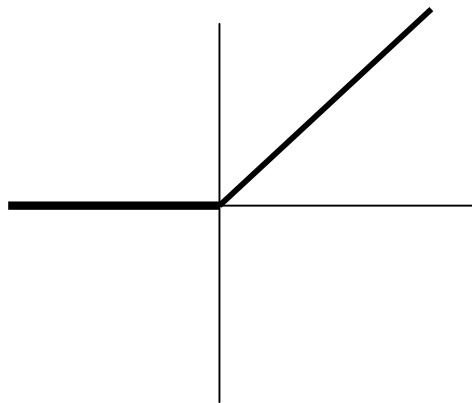


Пороговая,

Линейная,

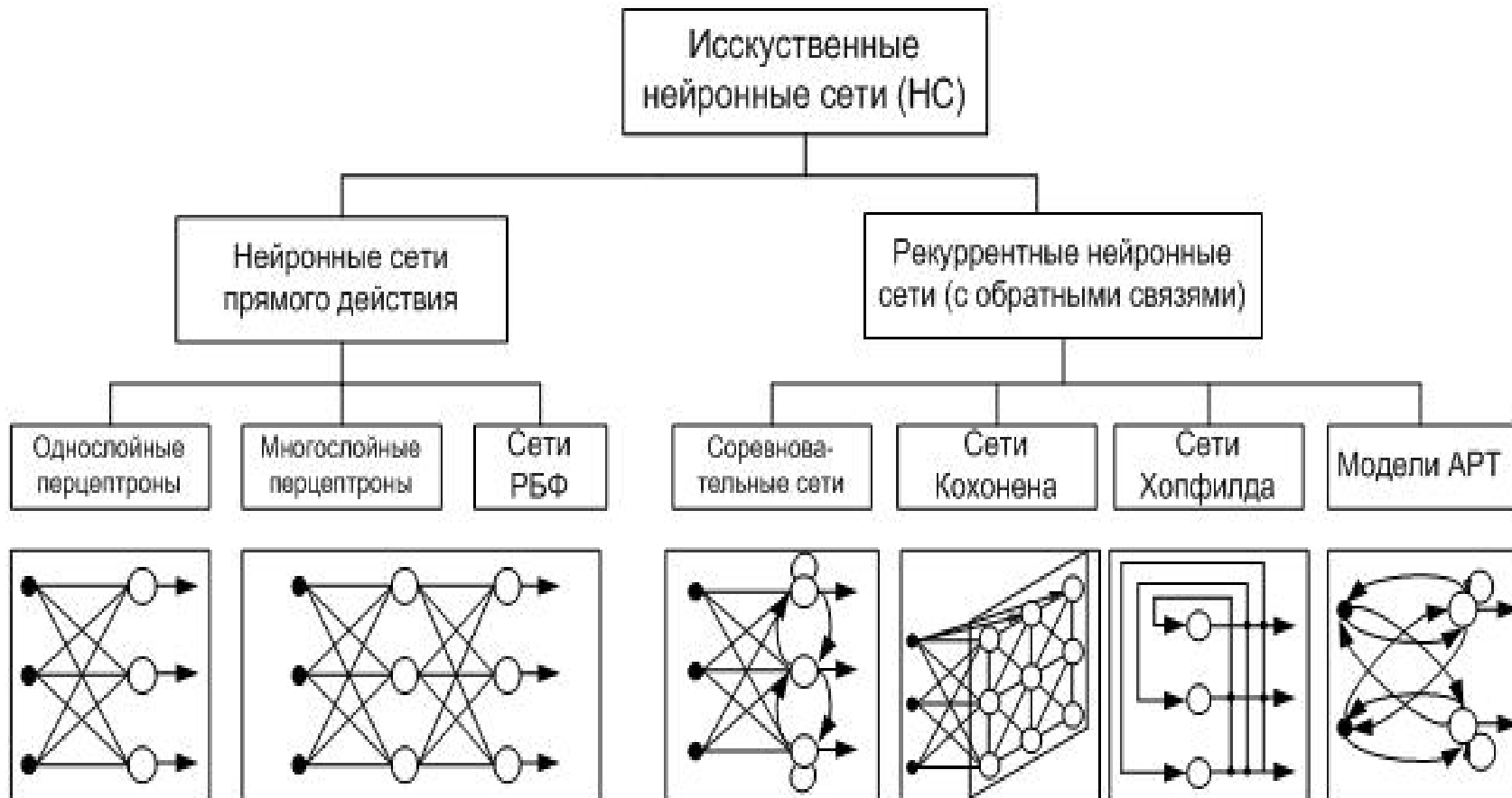
Сигмоидная,

Гауссовская



ReLU - Rectified Linear Unit

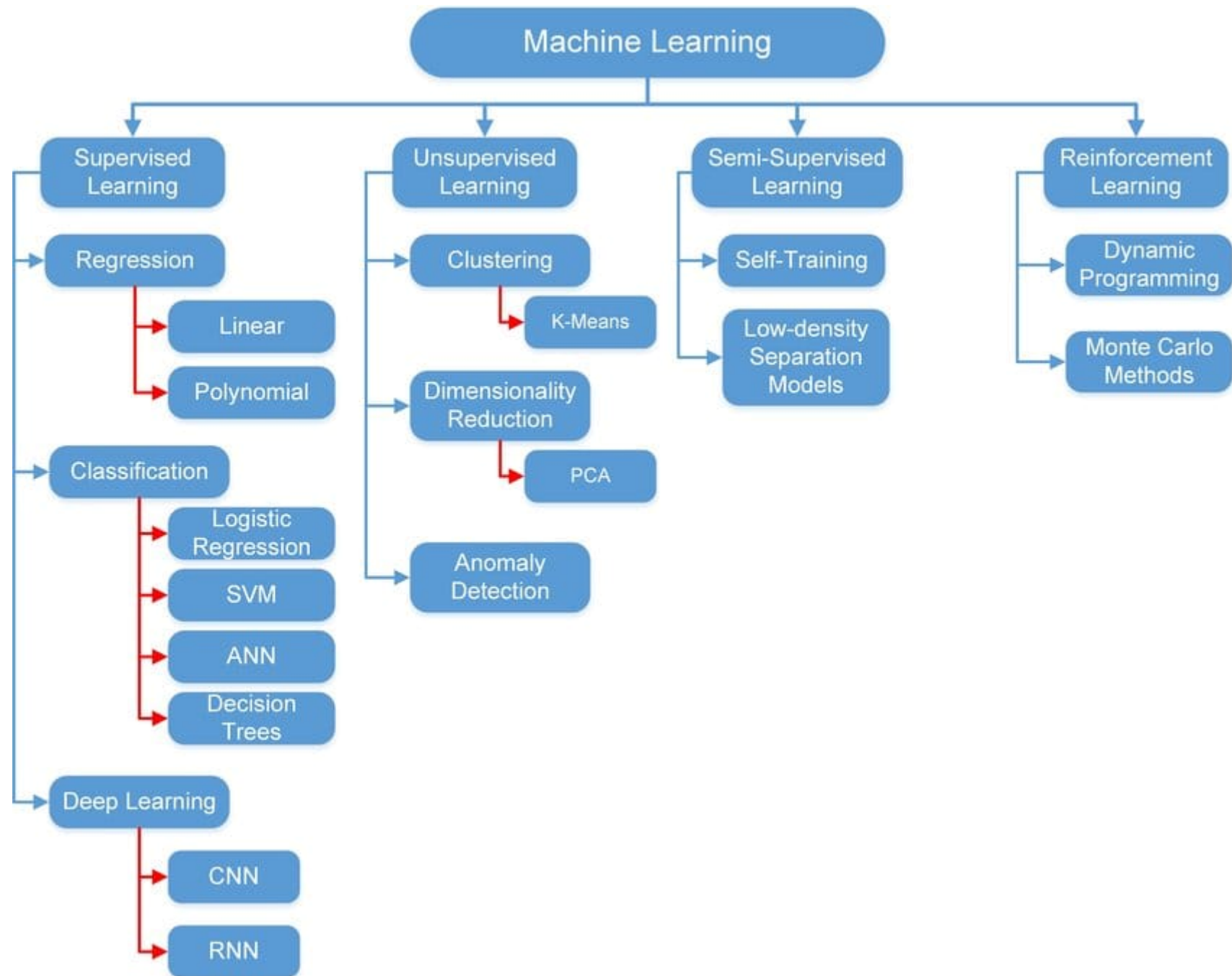
Классификация классических нейронных сетей



Еще: растущие НС, импульсные, сверточные, генеративные.

Типы обучения нейронных сетей

- С учителем (supervised learning)
 - Пример – входной и выходной (метка) векторы
- Без учителя (unsupervised learning)
 - Пример – входной вектор
- С частичным использованием учителя (semi-supervised learning)
 - Примеры размеченные и нет
- С поощрением и наказанием (Reinforcement learning) или обучение поведению
 - Поведение (выход) оценивается и НС получает сигнал поощрения или наказания (от среды или «критика» внутри системы)



Литература

1. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных [Электронный ресурс], 2015. 400 с.
2. Бринк Х. Машинное обучение [Электронный ресурс] / Х. Бринк, Д. Ричардс, М. Феверолф, 2017. 336 с.
3. Хайкин, Саймон. Нейронные сети [Текст] : полный курс / С. Хайкин, 2006. 1103 с.
4. Элбон, Крис. Машинное обучение с использованием Python. Сборник рецептов [Текст] : [пер. с англ.] / К. Элбон, 2020. 369 с.
5. М.Харрисон. Машинное обучение. Карманный справочник. - Диалектика, 2020.
6. А.Мюллер, С.Гвидо. Введение в машинное обучение с помощью Python. - 2017.
7. Бурков, А. Инженерия машинного обучения. - Москва : ДМК Пресс, 2022.
8. Лимановская, О. В. Основы машинного обучения : учебное пособие / О. В. Лимановская, Т. И. Алферьева. — 2-е изд. — Москва : ФЛИНТА, 2022.
9. Араки, М. Манга: Машинное обучение. — Москва : ДМК Пресс, 2020.
10. Гладилин, П. Е. Технологии машинного обучения : учебно-методическое пособие / П. Е. Гладилин, К. О. Боченина. — Санкт-Петербург : НИУ ИТМО, 2020.
11. Данилов, В. В. Нейронные сети : учебное пособие / В. В. Данилов. — Донецк : ДонНУ, 2020.
12. Ростовцев, В. С. Искусственные нейронные сети / В. С. Ростовцев. — 3-е изд., стер. — Санкт-Петербург : Лань, 2023.
13. Семериков, А. В. Классификация объектов на основе нейронной сети и методами дерева решения и ближайших соседей : учебное пособие / А. В. Семериков, М. А. Глазырин. — Ухта : УГТУ, 2022.
14. Нейронные сети - https://propoprogs.ru/neural_network
15. Машинное обучение - <https://propoprogs.ru/ml>
16. Шолле Ф. Глубокое обучение с R и Keras. - М.: ДМК Пресс, 2022.
17. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow. - СПб.: ООО "Диалектика", 2020.
18. Джулли А., Пал С. Библиотека Keras – инструмент глубокого обучения. - М.: ДМК Пресс, 2018.