

1. Термин Машинное обучение. Задачи, решаемые с помощью машинного обучения.

Машинное обучение - Machine Learning(ML)

Множество математических, статистических и вычислительных методов для разработки алгоритмов, способных решить задачу не прямым способом, а на основе поиска закономерностей в разнообразных входных данных. Процесс поиска закономерностей называют обучением.

Различают два типа обучения:

- **Обучение по прецедентам**, или **индуктивное обучение** - основано на выявлении эмпирических закономерностей в данных.
- **Дедуктивное обучение** - предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

Классические задачи, решаемые с помощью машинного обучения

- Классификация (*classification*)
- Кластеризация (*clustering/cluster analysis*)
- Регрессия (*regression*)
- Прогнозирование (*forecasting*)
- Поиск субоптимальных решений или стратегий
- Понижение размерности (*dimensionality reduction*)
- Визуализация данных (*data visualization*)
- Восстановление плотности распределения вероятности
- Поиск ассоциативных правил (*association rules learning*)
- Обнаружение аномалий / фильтрация выбросов (*outliers detection*)
- Одноклассовая классификация / Идентификация



2. Обучение по прецедентам (индуктивное обучение). Формальная постановка задачи.

Обучение по прецедентам (индуктивное обучение)

Дано конечное множество **прецедентов** (объектов) $\{X_1, \dots, X_m\} \subset X$, по каждому из которых собраны некоторые данные (описание), а также соответствующие им выходные значения (результаты) из множества допустимые ответов Y .

Совокупность всех имеющихся описаний прецедентов называется **обучающей выборкой (training sample)**, которая обычно является случайной выборкой объектов из генеральной совокупности Ω .

Обучающая выборка имеет вид $S^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$, где:

- пары «объект–ответ» (X_i, Y_i) – прецедент S_i^m из выборки S^m ;
- X_i – описание i -ого объекта из S^m , $i = 1, \dots, m$;
- Y_i – значение переменной Y для i -ого объекта из S^m , $i = 1, \dots, m$;
- m – число объектов в S^m .

Существует неизвестная **целевая функция (Target function)** $t^*: X \rightarrow Y$, значения которой известны только для конечного множества объектов обучающей выборки.

Необходимо построить алгоритм, который выдаст достаточно точный результат (выходное значение) для любых возможных входов (наборов значений признаков), в том числе таких, которые ещё не наблюдались.

Постановка задачи обучения

Дано:

- Обучающая выборка $S^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$;
- Предполагается, что существует некоторая неизвестная зависимость $t^*: X \rightarrow Y$ (**целевая функция**).

Задача обучения по прецедентам:

По обучающей выборке S^m построить некоторую **решающую функцию (decision function)** $a: X \rightarrow Y$, которая приближала бы целевую функцию t^* , причём не только на объектах обучающей выборки, но и на всём множестве возможных объектов генеральной совокупности Ω .

3. Обучение по прецедентам (индуктивное обучение). Общий порядок действий.

Общий порядок действий при обучении по прецедентам

- Анализ постановки задачи и исходных данных;
- Формулировка решения на математическом языке (задача формализуема, а результаты работы модели могут быть проверены);
- Предобработка данных и выделение ключевых признаков;
- Выбирается и фиксируется модель восстанавливаемой зависимости;
- 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;
- Эксплуатация модели при достижении требуемого качества, либо возврат к одному из предыдущих шагов (перенастройка модели, добыча новых данных и т. п.).

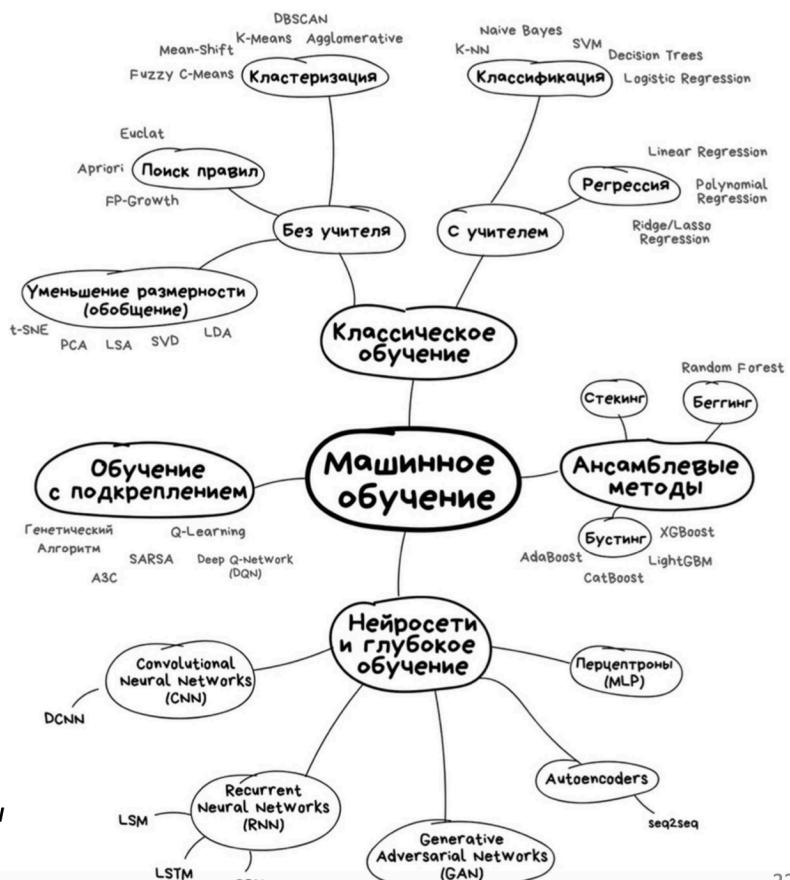
Можно сказать, что машинное обучение реализует подход **Case Based Reasoning (CBR)** — метод решения проблем рассуждением по аналогии, путем предположения на основе подобных случаев (прецедентов).

4. Типы машинного обучения.

- Классическое обучение:
 - Обучение с учителем (supervised learning)
 - Обучение без учителя (unsupervised learning)
- Обучение с подкреплением (reinforcement learning)
- Ансамблевые методы (Ensemble of models)
- Нейронные сети и глубокое обучение

Дополнительно выделяют:

- Частичное обучение (semi-supervised learning)
- Трансдуктивное обучение (transductive learning)
- Динамическое обучение (online learning)
- Активное обучение (active learning)
- Метаобучение (meta-learning или learning-to-learn)



5. Свойства алгоритма обучения. Обобщающая способность (generalization ability). Проблема недообучения и переобучения.

Свойства алгоритма обучения

Алгоритм обучения принимает на входе конечную обучающую выборку прецедентов и настраивает модель. Настроенная (обученная) модель затем используется для предсказания будущих прецедентов. Алгоритм должен обладать свойством **обучаемости** в следующих двух смыслах.

- Во-первых, алгоритм обучения должен обладать способностью к **обобщению** данных. Построенная им модель должна выдавать в среднем достаточно точные предсказания будущих прецедентов, т.е. обобщение определяет адекватный отклик на данные, выходящие за пределы имеющейся обучающей выборки. Оценки обобщающей способности, как правило, основываются на гипотезе, что прошлые и будущее прецеденты поступают случайно и независимо из одного и того же неизвестного вероятностного распределения. Эта гипотеза позволяет применить статистические методы для получения верхних оценок ожидаемой в будущем ошибки.
- Во-вторых, процесс обучения должен завершиться за приемлемое время. Обычно исследуется вопрос, является ли время обучения модели полиномиальным или экспоненциальным по длине выборки. Таким образом, проблематика вычислительного обучения тесно связана также и с вопросами **вычислительной сложности** алгоритмов.

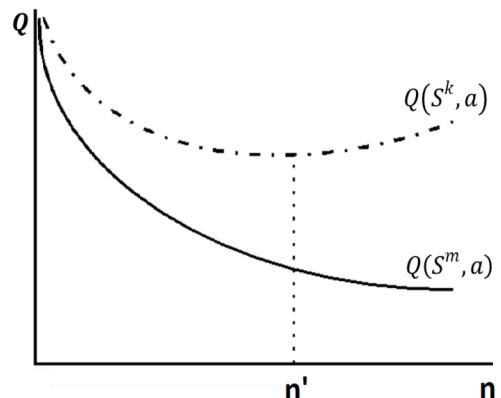
Обобщающая способность (generalization ability)

Обобщающая способность – это свойство модели отражать исходные данные в требуемые результаты ($X \rightarrow Y$) на всем множестве возможных объектов генеральной совокупности (во всех сценариях, а не только на тренировочных примерах).

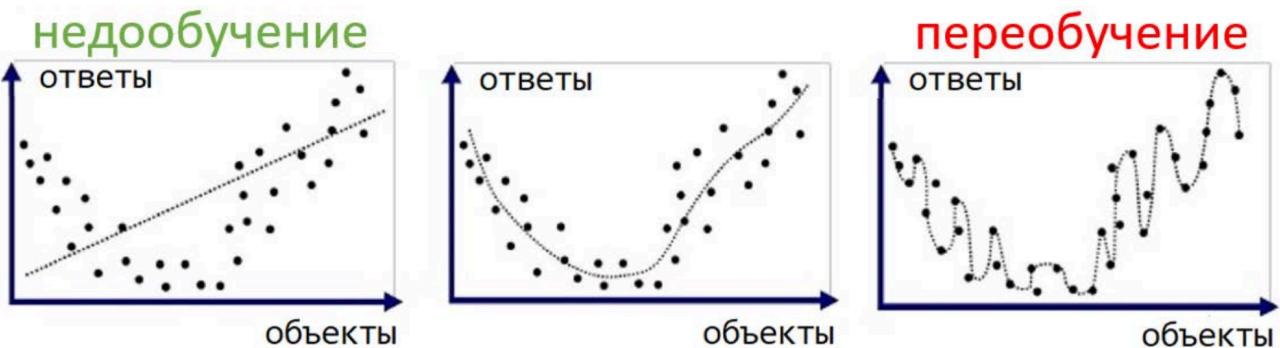
Если минимум функционала эмпирического риска (функционал качества) $Q(S^m, a)$ достигается на алгоритме a , то это не гарантирует, что a хорошо приближает целевую зависимость на произвольной контрольной выборке $S^k = (X'_i, Y'_i)_{i=1}^k$. **Обобщающая способность (generalization ability)** метода μ характеризуется величиной $Q(S^k, a) = Q(S^k, \mu(S^m))$ при условии, что выборки S^k и S^m являются представительными.

Метод обучения μ называется состоятельным, если при заданных достаточно малых значениях ε и η для любых простых выборок S^k и S^m справедлива оценка $Q(S^k, \mu(S^m)) \leq \varepsilon$ с вероятностью не менее $1 - \eta$.

Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте **переобучения (overtraining)** или **переподгонки (overfitting)**.



Проблема недообучения и переобучения



Недообучение (underfitting) возникает в том случае, когда модель слишком проста и содержит недостаточное число параметров n .

Переобучение (overfitting) возникает в том случае, когда модель слишком сложная и содержит избыточное число параметров n .

Как бороться:

- Уменьшить число настраиваемых параметров модели;
- По возможности увеличить число обучающих примеров;
- Уменьшить число итераций алгоритма обучения;
- Использовать эмпирические оценки обобщающей способности.

6. Свойства алгоритма обучения. Обобщающая способность (generalization ability).

Эмпирические оценки обобщающей способности

Свойства алгоритма обучения

Алгоритм обучения принимает на входе конечную обучающую выборку прецедентов и настраивает модель. Настроенная (обученная) модель затем используется для предсказания будущих прецедентов. Алгоритм должен обладать свойством **обучаемости** в следующих двух смыслах.

- Во-первых, алгоритм обучения должен обладать способностью к **обобщению** данных. Построенная им модель должна выдавать в среднем достаточно точные предсказания будущих прецедентов, т.е. обобщение определяет адекватный отклик на данные, выходящие за пределы имеющейся обучающей выборки. Оценки обобщающей способности, как правило, основываются на гипотезе, что прошлые и будущее прецеденты поступают случайно и независимо из одного и того же неизвестного вероятностного распределения. Эта гипотеза позволяет применить статистические методы для получения верхних оценок ожидаемой в будущем ошибки.
- Во-вторых, процесс обучения должен завершиться за приемлемое время. Обычно исследуется вопрос, является ли время обучения модели полиномиальным или экспоненциальным по длине выборки. Таким образом, проблематика вычислительного обучения тесно связана также и с вопросами **вычислительной сложности** алгоритмов.

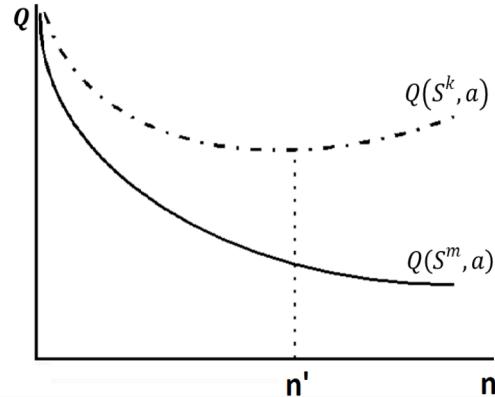
Обобщающая способность (*generalization ability*)

Обобщающая способность – это свойство модели отражать исходные данные в требуемые результаты ($X \rightarrow Y$) на всем множестве возможных объектов генеральной совокупности (во всех сценариях, а не только на тренировочных примерах).

Если минимум функционала эмпирического риска (функционал качества) $Q(S^m, a)$ достигается на алгоритме a , то это не гарантирует, что a хорошо приближает целевую зависимость на произвольной контрольной выборке $S^k = (X'_i, Y'_i)_{i=1}^k$. **Обобщающая способность (*generalization ability*)** метода μ характеризуется величиной $Q(S^k, a) = Q(S^k, \mu(S^m))$ при условии, что выборки S^k и S^m являются представительными.

Метод обучения μ называется состоятельным, если при заданных достаточно малых значениях ε и η для любых простых выборок S^k и S^m справедлива оценка $Q(S^k, \mu(S^m)) \leq \varepsilon$ с вероятностью не менее $1 - \eta$.

Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте **переобучения** (*overtraining*) или **переподгонки** (*overfitting*).



Эмпирические оценки обобщающей способности

Пусть дана выборка $S^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$. Разобьём её N различными способами на две непересекающиеся подвыборки – обучающую S_n^ℓ длины ℓ и контрольную S_n^k длины $k = m - \ell$. Для каждого разбиения $n=1, \dots, N$ построим алгоритм $a_n = \mu(S_n^\ell)$ и вычислим значение функционала эмпирического риска $Q_n = Q(S_n^k, a_n)$.

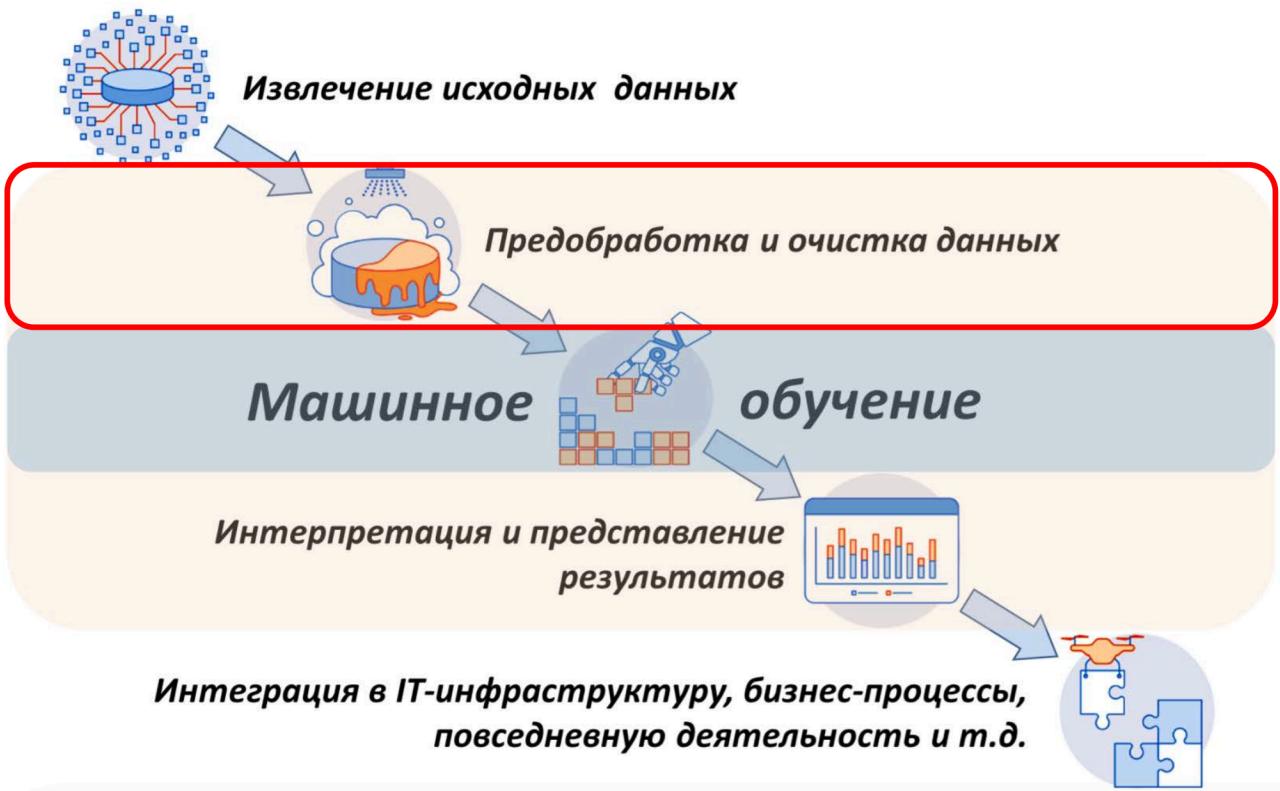
Среднее арифметическое значений Q_n по всем разбиениям называется оценкой скользящего контроля (*cross-validation, CV*):

$$CV(\mu, S^L) = \frac{1}{N} \sum_{n=1}^N Q(S_n^k, \mu(S_n^\ell))$$

Стандартом «де факто» считается методика $t \times q$ -кратного скользящего контроля (*$t \times q$ -fold cross-validation*), когда выборка случайным образом разбивается на q блоков равной (или почти равной) длины, каждый блок по очереди становится контрольной выборкой, а объединение всех остальных блоков – обучающей выборкой. Выборка S^m по-разному t раз разбивается на q блоков. Итого получается $N = t \times q$ разбиений.

7. Исходные данные. Признаковое описание объекта. Типы признаков (Атрибутов). Предобработка.

Предобработка и очистка данных



Типы признаков (Атрибутов)

Значения в исходном признаковом описании **прецедентов** (объектов) могут подразделяться на **количественные и качественные**.

Количественным (quantitative) называется признак, который имеет числовое представление и они могут быть:

- **дискретными (discrete data)** - выражаемые ограниченным набором значений (обычно целыми числами)
- **непрерывными (continuaes data)** - принимающие значения на непрерывной шкале значений.

Качественные (attribute, qualitative) признаки выражаются нечисловыми значениями и подразделяются на:

- **альтернативные (бинарные)** – имеют только два варианта значений.
- **атрибутивные (неупорядоченные/категориальные)** - имеет более двух вариантов, которые при этом выражаются в виде понятий или наименований.
- **порядковые (ординальные)** - имеют несколько ранжированных, т.е. упорядоченных по возрастанию или убыванию, качественных вариантов.

В результате, после предобработки и кодирования, в зависимости от множества допустимых значений D_f признаки делятся на следующие типы:

- **бинарный признак:** $D_f = \{0, 1\}$;
- **номинальный признак:** D_f — конечное множество;
- **порядковый признак:** D_f — конечное упорядоченное множество;
- **количественный признак:** D_f — множество действительных чисел.

Признаковое описание объектов (feature vector)

Признак (*feature*) — результат измерения некоторой характеристики объекта, т.е. отображение: $f: X \rightarrow D_f$, где D_f — множество допустимых значений признака.

Для всех прецедентов (объектов) выборки фиксируется совокупность из n признаков. Если для объекта X_i заданы признаки f_1, \dots, f_n , то вектор $(f_1(X_i), \dots, f_n(X_i)) = (x_i^1, \dots, x_i^n)$ называется признаковым описанием объекта $X_i \in X$.

Признаковое описание объекта (*feature vector*) — это вектор, который составлен из значений, соответствующих фиксированному набору признаков (характеристик) для данного объекта.

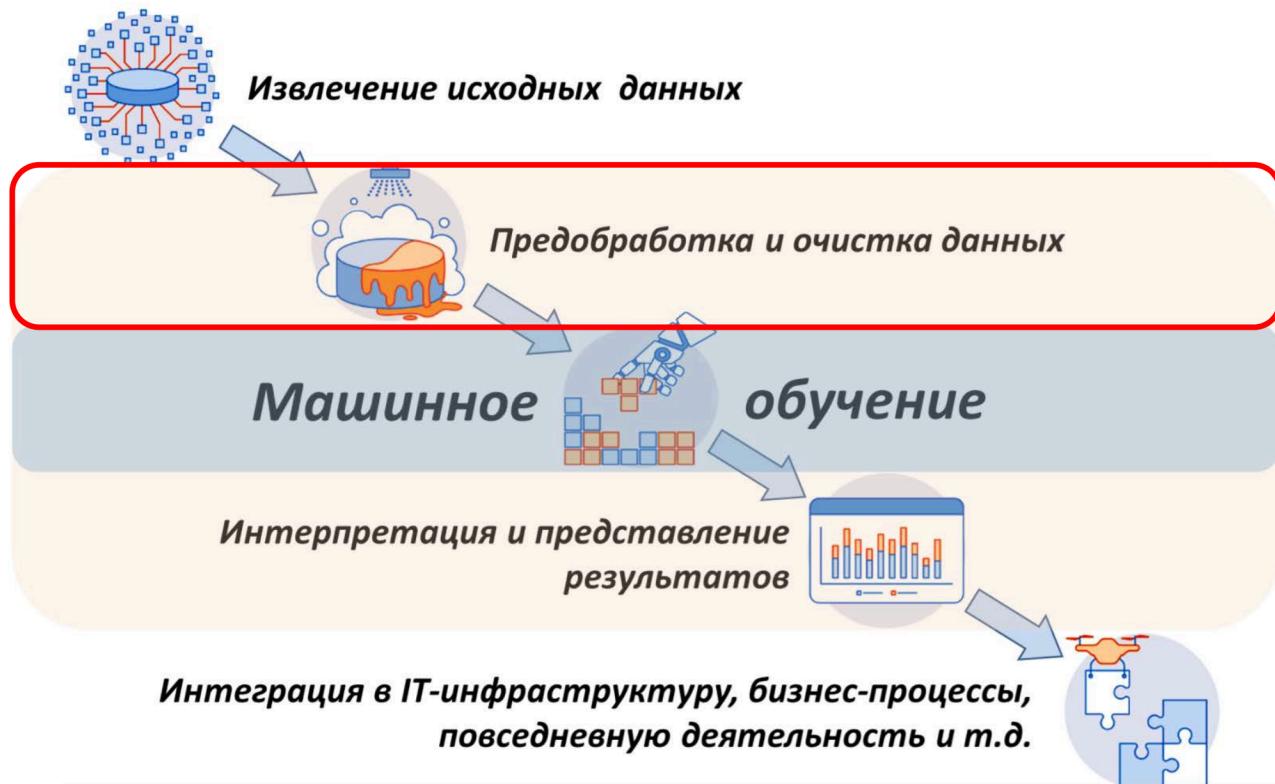
В машинном обучении признаковые описания допустимо отождествлять с самими объектами. При этом все множество X называют признаковым пространством.

Матрицей объектов-признаков (матрицей/таблицей исходных данных) называется совокупность признаковых описаний объектов обучающей выборки S^m длины m , записанной в виде матрицы размера $m \times n$ (m строк, n столбцов). Столбцы соответствуют признакам, строки — признаковым описаниям объектов.

	f_1	...	f_n
X_1	x_1^1	...	x_1^n
...
X_m	x_m^1	...	x_m^n

8. Исходные данные. Возможные проблемы исходных данных. Предобработка.

Предобработка и очистка данных



Возможные проблемы исходных данных

- Малый объем обучающей выборки;
- Некорректность входных данных;
 - Неполные
 - Неточные
- Противоречивость данных;
- Разнородность признаков;
- Неструктурированные/отсутствует разметка.

Риски, связанные с постановкой задачи:

- «грязные» данные – заказчик не обеспечивает качество данных;
- Неясные критерии качества модели – заказчик не определился с целями или индикаторами.

Учет пропусков

- Исключение объектов/признаков, имеющих неполные сведения (удаление строк/столбцов);
- Заполнить при помощи интерполяции;
- Найти в других источниках и тем самым дополнить данные;
- Закодировать пропуски специальным значением;
- Привлечь эксперта в соответствующей предметной области:
 - использование специфичных математических моделей
 - генерация псевдослучайных значений, подчиняющиеся некому распределению с учетом других известных признаков;
 - генерация синтетических данных.

Увеличение информативности примеров для повышения скорости и эффективности обучения

Еще одной целью предобработки данных, является увеличение информативности примеров для повышения скорости и эффективности обучения. Чем больше бит информации принесет каждый пример, тем лучше используются имеющиеся данные. Среднее количество информации, приносимой каждым примером x , равно энтропии распределения значений этой компоненты $H(x)$. Если эти значения сосредоточены в относительно небольшой области единичного интервала, информационное содержание такой компоненты мало и когда все значения переменной совпадают, эта переменная не несет никакой информации. Напротив, если значения переменной x равномерно распределены в единичном интервале, информация такой переменной максимальна.

Таким образом, общий принцип предобработки данных состоит в таком кодировании и нормировке непротиворечивых данных, чтобы добиться максимизации энтропии входов и выходов.

Нормировка данных

Стандартизация данных – это процесс приведения вектора каждого признака к такому виду, что его математическое ожидание станет нулевым, а дисперсия – единичной.

Нормализация данных – это процесс масштабирования вектора каждого признака, то есть приведение его к такому виду, что вектор будет иметь единичную норму (при этом есть разные способы оценки\подсчета нормы).

Линейное преобразование:

- $\tilde{x}_i = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}$ в единичный отрезок: $\tilde{x}_i \in [0,1]$
- $\tilde{x}_i = 2 \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}} - 1$ для отображения данных в интервал $[-1,1]$

L1 норма: $x'_i = \frac{x_i}{\|x\|_1} = \frac{x_i}{\sum_j |x_j|}$, где $\|x\|_1$ есть L1 норма, а вся формула целиком отображает процесс нормализации вектора x .

L2 норма: $x'_i = \frac{x_i}{\|x\|_2} = \frac{x_i}{\sqrt{\sum_j x_j^2}}$, где $\|x\|_2$ есть L2 норма, а вся формула целиком отображает процесс нормализации вектора x .

9. Случайная природа входных данных. Вероятностные характеристики выборки.

Основные распределения значений признаков.

Вероятностные характеристики выборки

$X = \{x_1 \dots x_n\}$ – изучаемая выборка

- **Медиана** – число, характеризующее выборку по среднему из ее значений. То есть, если все данные выборки различны, и она упорядочена по возрастанию, то ровно половина из элементов выборки будет меньше медианы, и ровно половина – больше. x_k – медиана, если $x_j < x_k$, при $j \in [1, k)$ и $x_k < x_i$, при $i \in (k, n]$ и $k=n/2$.
- **Математическое ожидание** – среднее значение вероятностных элементов выборки. Формально она рассчитывается так:

$$M|X| = \sum x_i p_i$$

- **Выборочное среднее** (несмешённая оценка мат. ожидания $E[X]$).

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

- **Дисперсия** – мера разброса элементов выборки относительно ее математического ожидания. Рассчитывается данная величина по формуле

$$D|X| = \sum (x_i - M|X|)^2 p_i$$

- **Выборочная дисперсия и среднеквадратическое отклонение** – величина, характеризующая рассеивание значений выборки относительно ее математического ожидания. Формула расчета

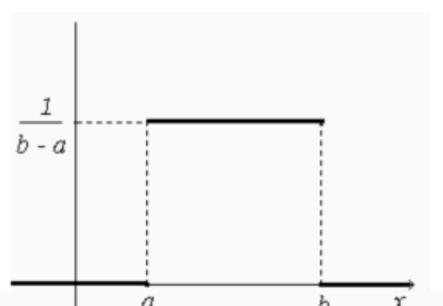
$$\sigma = \sqrt{\frac{1}{n} \sum (x_i - \bar{x})^2}$$

Непрерывное равномерное распределение

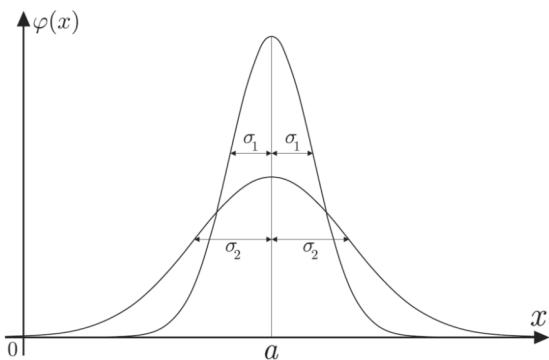
Непрерывное равномерное распределение в теории вероятностей – распределение случайной вещественной величины, принимающей значения, принадлежащие некоторому промежутку конечной длины, характеризующееся тем, что плотность вероятности на этом промежутке почти всюду постоянна.

Говорят, что случайная величина имеет непрерывное равномерное распределение на отрезке $[a, b]$, где $a, b \in \mathbb{R}$, если ее плотность $f_X(x)$ имеет вид:

$$f_X(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \\ 0, & x \notin [a, b] \end{cases}$$



Нормальное распределение



Определение. Случайная величина ξ имеет нормальное распределение вероятностей с параметрами μ и σ^2 , если ее Закон плотности распределения вероятности задается формулой:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

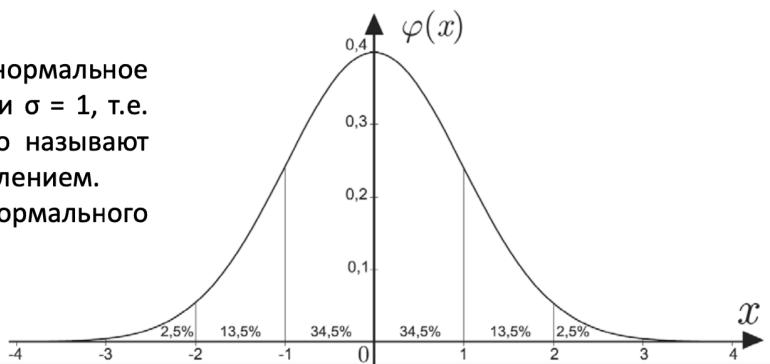
где σ — среднеквадратическое отклонение;
 μ — математическое ожидание.

Краткое обозначение: $\xi \sim N(\mu, \sigma^2)$

Особую роль играет нормальное распределение с параметрами $\mu = 0$ и $\sigma = 1$, т.е. распределение $N(0, 1)$, которое часто называют *стандартным* нормальным распределением.

Плотность стандартного нормального распределения:

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



Распределение хи-квадрат

Определение. Пусть случайные величины $\xi_1, \xi_2, \dots, \xi_n$ — независимы, и каждое из них имеет стандартное нормальное распределение $N(0, 1)$. Говорят, что случайная величина χ_n^2 определенная как:

$$\chi_n^2 = \xi_1^2 + \dots + \xi_n^2$$

имеет распределение хи-квадрат с n степенями свободы.

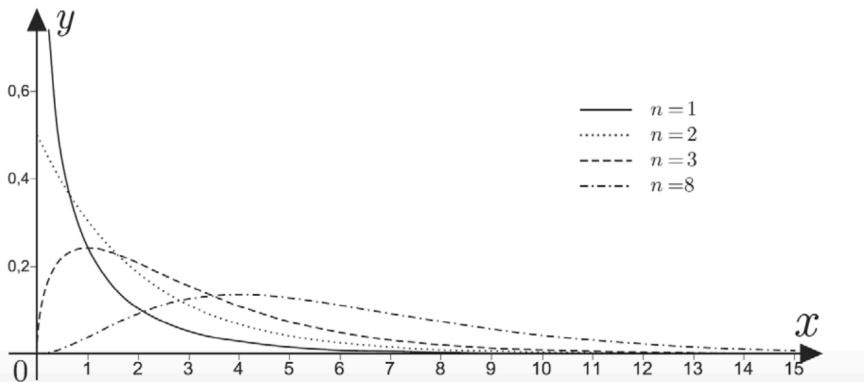
Функция плотности χ_n^2 в точке x ($x > 0$) равна:

$$\frac{1}{2^{n/2}} \frac{1}{\Gamma(n/2)} x^{\frac{n}{2}-1} e^{-x/2}$$

где $\Gamma(\cdot)$ есть гамма функция. На практике эта плотность распределения непосредственно используется редко.

Свойства. Математическое ожидание и дисперсия случайной величины χ_n^2 равны:

$$M\chi_n^2 = n, D\chi_n^2 = 2n.$$



Распределение Стьюдента

Определение. Пусть случайные величины $\xi_0, \xi_1, \dots, \xi_n$ – независимы, и каждое из них имеет стандартное нормальное распределение $N(0,1)$.

Введем случайную величину: $t_n = \frac{\xi_0}{\sqrt{\frac{1}{n} \sum_{i=1}^n \xi_i^2}}$

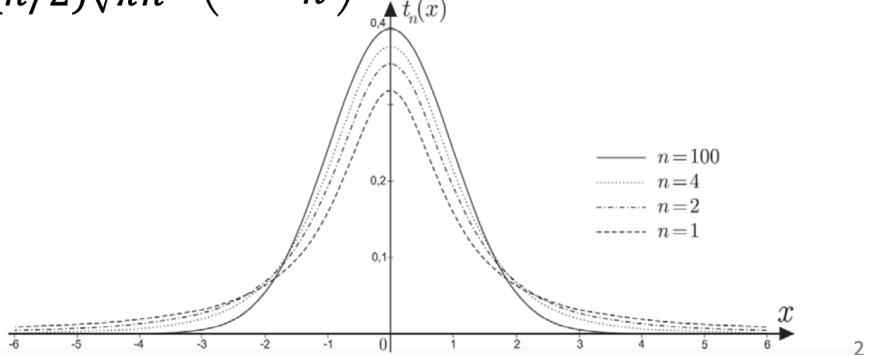
Ее распределение называют распределением Стьюдента. Саму случайную величину часто называют стьюдентовской дробью или стьюдентовым отношением. Число $n = 1, 2, \dots$ называют числом степеней свободы распределения Стьюдента.

Плотность распределения Стьюдента в точке x равна:

$$\frac{\Gamma((n+1)/2)}{\Gamma(n/2)\sqrt{\pi n}} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Свойства:

$$Mt_n = 0, Dt_n = \frac{n}{n-2}.$$



F-распределение

Определение. Пусть $\eta_1, \dots, \eta_m; \xi_1, \dots, \xi_n$ (где m, n – натуральные числа) обозначают независимые случайные величины, каждое из которых распределено по стандартному нормальному закону распределения $N(0,1)$. Говорят, что случайная величина $F_{m,n}$, определенная как

$$F_{m,n} = \frac{\frac{1}{m}(\eta_1^2 + \dots + \eta_m^2)}{\frac{1}{n}(\xi_1^2 + \dots + \xi_n^2)}$$

имеет F-распределение с параметрами m и n . Натуральные числа m, n называют числами степеней свободы. F-распределение иногда называют еще распределением дисперсионного отношения.

Свойства.

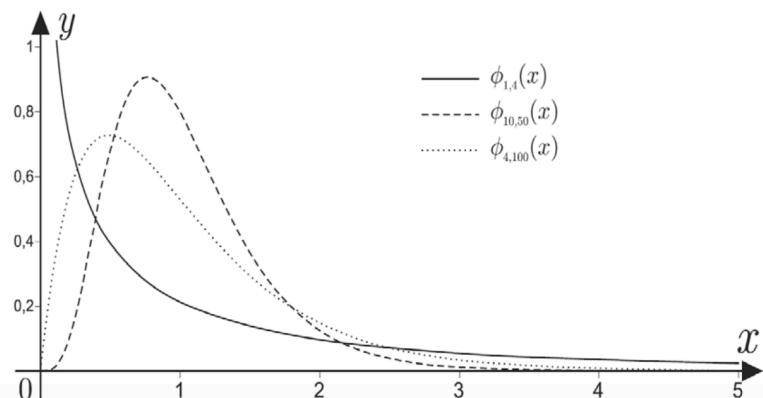
Математическое ожидание

для $n > 2$:

$$MF_{m,n} = \frac{n}{n-2}$$

Дисперсия для $n > 4$:

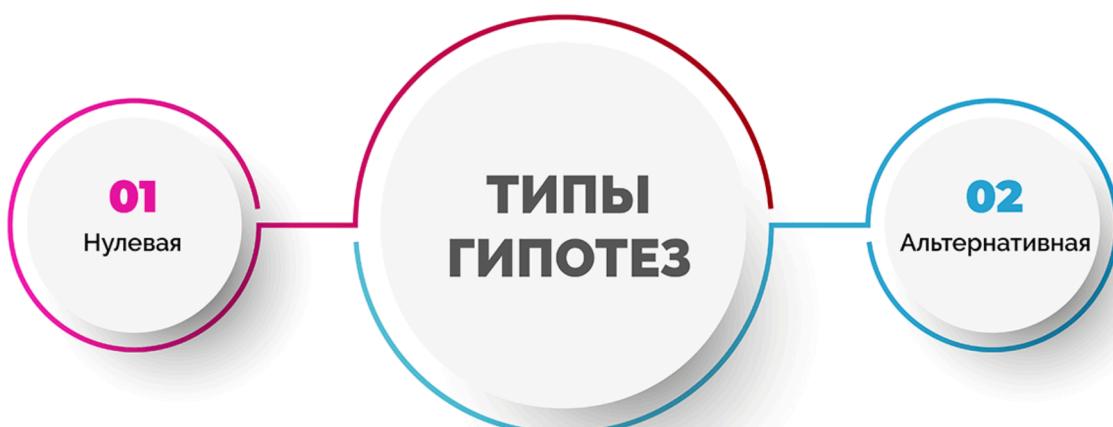
$$DF_{m,n} = \frac{2n^2(m+n-2)}{m(n-2)^2(n-4)}$$



10. Статистические гипотезы и их проверка. Статистическая значимость.

Статистические гипотезы

Статистическая гипотеза — гипотеза о виде распределения и свойствах случайной величины, которые можно подтвердить или опровергнуть применением статистических методов к данным выборки.



Нулевая гипотеза — принимаемое по умолчанию предположение о том, что не существует связи между двумя наблюдаемыми событиями, феноменами. Часто в качестве нулевой гипотезы выступают предположения об отсутствии взаимосвязи или корреляции между исследуемыми переменными, об отсутствии различий (однородности) в распределениях (параметрах распределений) в двух и/или более выборках. Для обозначения нулевой гипотезы часто используют символ H_0 .

Проверка гипотез

Нулевая гипотеза H_0 считается верной, пока нельзя доказать обратное. Проверка фальсифицируемости нулевой гипотезы — общепринятый способ обеспечения строгости исследования. Если прямого способа проверки у нас нет, приходится прибегать к проверкам косвенным. **Статистика как наука даёт чёткие условия, при наступлении которых нулевая гипотеза может быть отвергнута.**

- Если некоторое явление логически неизбежно следует из гипотезы, но в природе не наблюдается, то это значит, что гипотеза неверна.
- Если происходит то, что при гипотезе происходить не должно, это тоже означает ложность гипотезы.

Заметим, что строго говоря, **косвенным образом доказать гипотезу нельзя, хотя опровергнуть — можно.**

При статистическом выводе исследователь пытается показать несостоятельность нулевой гипотезы, несогласованность её с имеющимися опытными данными, то есть отвергнуть гипотезу. При этом подразумевается, что должна быть принята другая, альтернативная (конкурирующая) гипотеза, исключающая нулевую гипотезу. Если же данные, наоборот, подтверждают нулевую гипотезу, то она не отвергается.

Проверка статистической значимости

Смысл статистической значимости заключается в том, чтобы определить, имеет ли под собой какое-то основание разница между двумя показателями, или же она случайна.

Выберем уровень вероятности $\epsilon > 0$. Условимся считать событие практически невозможным, если его вероятность меньше ϵ . Когда речь идет о проверке гипотез, число ϵ называют уровнем значимости.

Уровень значимости — это вероятность того, что мы сочли различия существенными, в то время как они на самом деле случайны. Уровень значимости показывает степень достоверности выявленных различий между выборками, т.е. показывает, насколько мы можем доверять тому, что различия действительно есть.

Уровни значимости: $p \leq 0,05$, $p \leq 0,01$, $p \leq 0,001$.

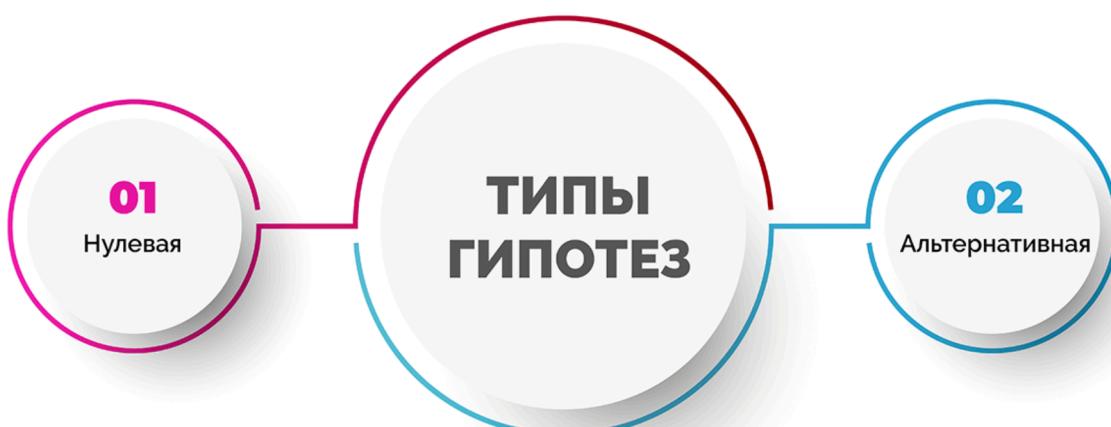
Определение. Событие А называется критическим для гипотезы Н, или критерием для Н. Если $P(A|H) \leq \epsilon$, то ϵ называют гарантированным уровнем значимости критерия А для Н.

Критерий для проверки гипотезы — это решающее правило, отвергающее или принимающее нулевую гипотезу на основе выборочных наблюдений.

11. Статистические гипотезы и их проверка. Ошибки первого и второго рода.

Статистические гипотезы

Статистическая гипотеза — гипотеза о виде распределения и свойствах случайной величины, которые можно подтвердить или опровергнуть применением статистических методов к данным выборки.



Нулевая гипотеза — принимаемое по умолчанию предположение о том, что не существует связи между двумя наблюдаемыми событиями, феноменами. Часто в качестве нулевой гипотезы выступают предположения об отсутствии взаимосвязи или корреляции между исследуемыми переменными, об отсутствии различий (однородности) в распределениях (параметрах распределений) в двух и/или более выборках. Для обозначения нулевой гипотезы часто используют символ H_0 .

Проверка гипотез

Нулевая гипотеза H_0 считается верной, пока нельзя доказать обратное. Проверка фальсифицируемости нулевой гипотезы — общепринятый способ обеспечения строгости исследования. Если прямого способа проверки у нас нет, приходится прибегать к проверкам косвенным. **Статистика как наука даёт чёткие условия, при наступлении которых нулевая гипотеза может быть отвергнута.**

- Если некоторое явление логически неизбежно следует из гипотезы, но в природе не наблюдается, то это значит, что гипотеза неверна.
- Если происходит то, что при гипотезе происходить не должно, это тоже означает ложность гипотезы.

Заметим, что строго говоря, **косвенным образом доказать гипотезу нельзя, хотя опровергнуть — можно.**

При статистическом выводе исследователь пытается показать несостоятельность нулевой гипотезы, несогласованность её с имеющимися опытными данными, то есть отвергнуть гипотезу. При этом подразумевается, что должна быть принята другая, альтернативная (конкурирующая) гипотеза, исключающая нулевую гипотезу. Если же данные, наоборот, подтверждают нулевую гипотезу, то она не отвергается.

Ошибки первого и второго рода

Возможны ошибки двух родов: первого рода (α) и второго рода (β).

Ошибка первого рода состоит в том, что гипотеза H_0 будет отвергнута, хотя на самом деле она правильная. Вероятность допустить такую ошибку называют **уровнем значимости** и обозначают буквой α («альфа»).

Ошибка второго рода состоит в том, что гипотеза H_0 будет принята, но на самом деле она неправильная. Вероятность совершить эту ошибку обозначают буквой β («бета»). Значение $(1-\beta)$ называют **мощностью критерия** — это вероятность отклонения ложной гипотезы, т.е. способность выявлять даже мелкие различия. Чем мощнее критерий, тем лучше он отвергает нулевую гипотезу.

В практических задачах, как правило, задают **уровень значимости**, и наиболее часто выбирают значения: $\alpha=0.1$, $\alpha=0.05$, $\alpha=0.01$.

Нулевая гипотеза H_0	Ложная	Истинная
Отклоняется	Ошибки нет ($1-\beta$)	Ошибка I рода (ложноположительный вывод), ложная тревога (α)
Принимается (не отклоняется)	Ошибка II рода (ложнотрицательный вывод), пропуск цели (β)	Ошибки нет ($1-\alpha$)

12. Методика проверки статистических гипотез. P-value. Доверительный интервал (Confidence interval).

Методика проверки статистических гипотез

Пусть задана случайная выборка $X^m = (x_1, \dots, x_m)$ — последовательность m объектов из множества X . Предполагается, что на множестве X существует некоторая неизвестная вероятностная мера P .

1. Формулируются *нулевая* H_0 и альтернативная H_1 гипотезы о распределении вероятностей на множестве X .
2. Задаётся некоторая статистика (функция выборки — на след. слайде) $T: X^m \rightarrow \mathbb{R}$, для которой в условиях справедливости гипотезы H_0 выводится функция распределения $F(T)$ и/или плотность распределения $p(T)$.
3. 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 $\alpha=0.05$.
4. На множестве допустимых значений статистики T выделяется *критическое множество* Ω_α наименее вероятных значений статистики T , такое, что $P\{T \in \Omega_\alpha | H_0\} = \alpha$.
5. 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.

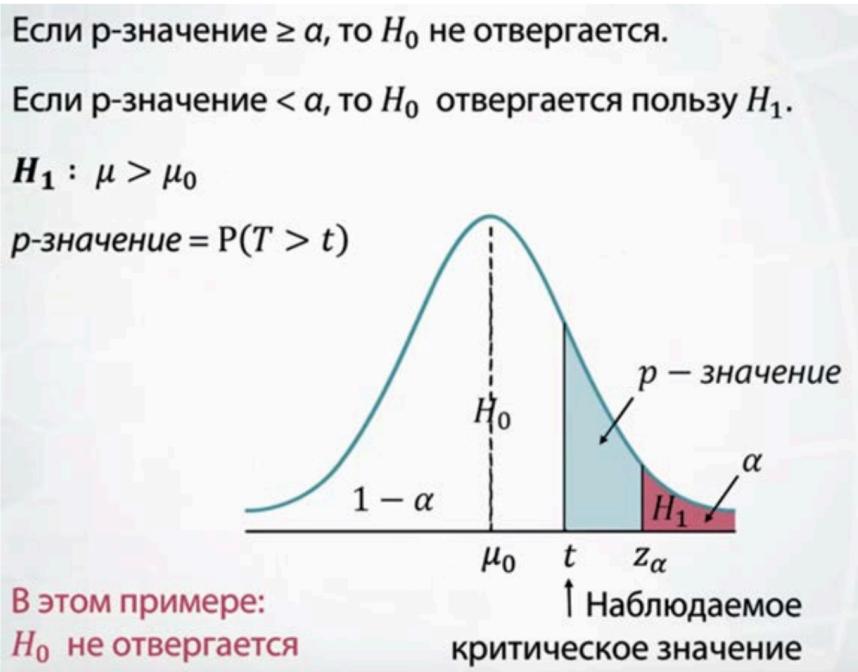
Важно! Если данные не противоречат нулевой гипотезе, это ещё не значит, что гипотеза верна.

Тому есть две причины:

- По мере увеличения длины выборки нулевая гипотеза может сначала приниматься, но потом выявляться 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.

P-value или р-значение

P-value или р-значение – одна из ключевых величин, используемых в статистике при тестировании гипотез. Она показывает вероятность получения наблюдаемых результатов при условии, что нулевая гипотеза верна, или вероятность ошибки в случае отклонения нулевой гипотезы.



Доверительный интервал (Confidence interval)

В математической статистике — интервал, в пределах которого с заданной вероятностью лежат выборочные оценки статистических характеристик генеральной совокупности.

Если оценку среднего требуется связать с **определенной вероятностью**, то интересующий параметр генеральной совокупности нужно оценивать не одним числом, а интервалом. Доверительным интервалом называют интервал, в котором с определённой вероятностью P находится значение оцениваемого показателя генеральной совокупности. Доверительный интервал, в котором с вероятностью $P = 1 - \alpha$ (α - квантиль стандартного нормального распределения) находится случайная величина \bar{X} , рассчитывается следующим образом:

$$\bar{X} - z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}}$$

где $z_{1-\frac{\alpha}{2}}$ - критическое значение стандартного нормального распределения для уровня значимости $\alpha = 1 - P$, которое можно найти в приложении к практически любой книге по статистике.

На практике среднее значение генеральной совокупности μ и дисперсия σ^2 не известны, поэтому дисперсия генеральной совокупности заменяется дисперсией выборки S^2 , а среднее генеральной совокупности - средним значением выборки \bar{X} . Таким образом, доверительный интервал в большинстве случаев рассчитывается так:

$$\bar{X} - t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{S}{\sqrt{n}} \leq \mu \leq \bar{X} + t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{S}{\sqrt{n}}$$

где S -несмешённое выборочное стандартное отклонение, имеет распределение Стьюдента с $n-1$ степенями свободы $t(n-1)$. Пусть $t_{\alpha, n-1}$ — α -квантили распределения Стьюдента.

Формулу доверительного интервала можно использовать для оценки среднего генеральной совокупности, если

- известно стандартное отклонение генеральной совокупности;
- или стандартное отклонение генеральной совокупности не известно, но объём выборки - больше 30.

13. Проверка гипотезы о математическом ожидании. Критерий согласия Пирсона Хи-квадрат (Chi-square test).

Проверка гипотезы о математическом ожидании

Может быть два варианта, когда дисперсия известна и когда неизвестна. Рассмотрим случай, когда она неизвестна.

Сравнение выборочного среднего с заданным значением

Задана выборка $x^m = (x_1, \dots, x_m)$, $x_i \in \mathbb{R}$, $x_i \sim N(\mu, \sigma^2)$.

Дополнительное предположение: выборка простая и нормальная.

Нулевая гипотеза $H_0: \bar{x} = \mu_0$ (выборочное среднее равно заданному числу μ_0), альтернативная $H_1: \bar{x} \neq \mu_0$

Статистика критерия:

$$t = \frac{(\bar{x} - \mu_0)}{\frac{s}{\sqrt{m}}}$$

имеет t-распределение Стьюдента (числитель-нормальное/знаменатель хи-квадрат) 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.

Критерий (при уровне значимости α):

Гипотеза H_0 принимается, если $|t| < t_{1-\alpha/2}$, в противном случае отвергается.

Критерий согласия Пирсона Хи-квадрат (Chi-square test)

Критерий согласия проверяет, согласуется ли заданная выборка с заданным фиксированным распределением (семейством распределений) или с другой выборкой, т.е. критерий согласия для проверки гипотезы о законе распределения.

Критерий хи-квадрат — любая статистическая проверка гипотезы, в которой выборочное распределение критерия имеет распределение хи-квадрат при условии верности нулевой гипотезы.

Пусть X — исследуемая случайная величина. Требуется проверить гипотезу H_0 о том, что данная случайная величина подчиняется закону распределения $F(x)$. Данна выборка $X^n = (x_1, \dots, x_n)$, $x_i \in [a, b]$, $\forall i = 1 \dots n$. Необходимо построить эмпирический закон распределения $F'(x)$ случайной величины X .

- Разделим $[a, b]$ на k непересекающихся интервалов $[a_i, b_i], i=1 \dots k$.
- Пусть n_j — количество наблюдений в j -м интервале;
- $p_j = F(b_j) - F(a_j)$ — вероятность попадания наблюдения в j -й интервал при выполнении гипотезы H'_0 ;
- $E_j = np_j$ — ожидаемое число попаданий в j -й интервал;
- тогда распределение Хи-квадрат с числом степеней свободы $k-1$ будет иметь следующую статистику (критерий Хи-квадрат Пирсона):

$$\chi^2 = \sum_{j=1}^k \frac{(n_j - E_j)^2}{E_j} \sim \chi^2_{k-1}$$

В зависимости от значения критерия χ^2 гипотеза H_0 может приниматься либо отвергаться:

$\chi^2 < \chi^2_1 < \chi^2_2$ — гипотеза H_0 выполняется.

$\chi^2 \geq \chi^2_2$ — попадает в правый «хвост» распределения, гипотеза H_0 отвергается.

14. Корреляционный анализ. Отбор признаков. Ограничения.

Корреляция (от лат. *correlatio* «соотношение, взаимосвязь»), или корреляционная зависимость — статистическая взаимосвязь двух или более случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми). При этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин.

Значительная корреляция между двумя случайными величинами всегда является свидетельством существования некоторой статистической связи в данной выборке, но эта связь не обязательно должна наблюдаваться для другой выборки и иметь причинно-следственный характер.

Отсутствие корреляции между двумя величинами ещё не значит, что между ними нет никакой связи. Например, зависимость может иметь сложный нелинейный характер, который корреляция не выявляет.

Некоторые виды коэффициентов корреляции могут быть положительными или отрицательными.

- В первом случае предполагается, что мы можем определить только наличие или отсутствие связи, а во втором — также и её направление.

Если предполагается, что на значениях переменных задано отношение строгого порядка, то в этом случае:

- **Отрицательная** корреляция — корреляция, при которой увеличение одной переменной связано с уменьшением другой.
- **Положительная** корреляция в таких условиях — это такая связь, при которой увеличение одной переменной связано с увеличением другой переменной.

Возможна также ситуация отсутствия статистической взаимосвязи — например, для независимых случайных величин.

Метод вычисления коэффициента корреляции зависит от вида шкалы, к которой относятся переменные. Так, для измерения переменных с интервальной и количественной шкалами необходимо использовать коэффициент корреляции Пирсона (корреляция моментов произведений).

Если, по меньшей мере, одна из двух переменных имеет порядковую шкалу, либо не является нормально распределённой, необходимо использовать ранговую корреляцию Спирмена или τ (тау) Кендалла.

Ограничения корреляционного анализа

- Применение возможно при наличии достаточного количества наблюдений для изучения.
- Коэффициент корреляции отражает «зашумлённость» линейной зависимости (верхняя строка)
- Коэффициент корреляции не описывает наклон линейной зависимости (средняя строка)
- Коэффициент корреляции совсем не подходит для описания сложных, нелинейных зависимостей (нижняя строка).
- Для распределения, показанного в центре рисунка, коэффициент корреляции не определен, так как дисперсия у равна нулю.

Отбор признаков на основе корреляции

Отбор признаков на основе меры корреляции (англ. *Correlation Feature Selection*, CFS) оценивает подмножество признаков на базе следующей гипотезы:

«Хорошие поднаборы признаков содержат признаки, сильно коррелирующие с классификацией, но не коррелирующие друг с другом».

Следующее равенство даёт оценку поднабора признаков S , состоящего из k признаков:

$$Merit_{S_k} = \frac{k\bar{r}_{cf}}{\sqrt{k + k(k - 1)\bar{r}_{ff}}}$$

В формулу входят среднее значение всех корреляций признак-классификация \bar{r}_{cf} , и среднее значение всех корреляций признак-признак \bar{r}_{ff} .

15. Регрессионный анализ (regression analysis). Цели и задачи. Математическое определение регрессии

Набор методов моделирования измеряемых данных и исследования их свойств, относится к разделам математической статистики и машинного обучения. Осуществляет исследование влияния одной или нескольких независимых переменных X_1, X_2, \dots, X_p на зависимую переменную Y .

Независимые переменные называют регрессорами, предикторами или объясняющими переменными, а зависимая переменная является результирующей, критериальной или регрессантом. Терминология зависимых и независимых переменных отражает лишь математическую зависимость переменных, а не причинно-следственные отношения.

Наиболее распространенный вид регрессионного анализа — линейная регрессия, когда находят линейную функцию, которая, согласно определённым математическим критериям, наилучшим образом соответствует данным.

Регрессионный анализ очень тесно связан с корреляционным анализом. В корреляционном анализе исследуется направление и теснота связи между количественными переменными. В регрессионном анализе исследуется форма зависимости между количественными переменными.

Регрессионный анализ используется для прогноза, анализа временных рядов, тестирования гипотез и выявления скрытых взаимосвязей в данных.

Цели и задачи регрессионного анализа

Цель регрессионного анализа – с помощью уравнения регрессии предсказать ожидаемое среднее значение результирующей переменной, т.е. определение степени детерминированности вариации критериальной (зависимой) переменной от независимых переменных (предикторов).

Основные задачи регрессионного анализа следующие:

- определения вида и формы зависимости;
- оценка параметров уравнения регрессии;
- проверка значимости уравнения регрессии;
- проверка значимости отдельных коэффициентов уравнения (вклад отдельных независимых переменных в вариацию зависимой);
- построение интервальных оценок коэффициентов;
- исследование характеристик точности модели;
- построение точечных и интервальных прогнозов результирующей переменной (предсказание значения зависимой переменной с помощью независимых).

Как и корреляционный анализ, регрессионный анализ отражает только количественные зависимости между переменными. Причинно-следственные зависимости регрессионный анализ не отражает. Гипотезы о причинно-следственной связи переменных должны формулироваться и обосновываться исходя из теоретического анализа содержания изучаемого явления.

Математическое определение регрессии

- Пусть Y, X_1, X_2, \dots, X_p — случайные величины с заданным совместным распределением вероятностей.
- Если для каждого набора значений $X_1 = x_1, X_2 = x_2, \dots, X_p = x_p$ определено условное математическое ожидание (**уравнение регрессии в общем виде**):

$$f(x_1, x_2, \dots, x_p) = E(Y|X_1 = x_1, X_2 = x_2, \dots, X_p = x_p),$$

то функция $f(x_1, x_2, \dots, x_p)$ называется регрессией величины Y по величинам X_1, X_2, \dots, X_p , а ее график – **линией регрессии**.

- Зависимость Y от X_1, X_2, \dots, X_p проявляется в изменении средних значений Y при изменении X_1, X_2, \dots, X_p .
- При каждом фиксированном наборе значений $X_1 = x_1, X_2 = x_2, \dots, X_p = x_p$ величина Y остаётся случайной величиной с определённым распределением.
- Для выяснения вопроса, насколько точно регрессионный анализ оценивает изменение Y при изменении X_1, X_2, \dots, X_p , используется средняя величина дисперсии Y при разных наборах значений X_1, X_2, \dots, X_p (фактически речь идет о мере рассеяния зависимой переменной вокруг линии регрессии).

16. Регрессионный анализ. Линейная и Нелинейная регрессия.

Набор методов моделирования измеряемых данных и исследования их свойств, относится к разделам математической статистики и машинного обучения. Осуществляет исследование влияния одной или нескольких независимых переменных X_1, X_2, \dots, X_p на зависимую переменную Y .

Независимые переменные называют регрессорами, предикторами или объясняющими переменными, а зависимая переменная является результирующей, критериальной или регрессантом. Терминология зависимых и независимых переменных отражает лишь математическую зависимость переменных, а не причинно-следственные отношения.

Наиболее распространенный вид регрессионного анализа — линейная регрессия, когда находят линейную функцию, которая, согласно определённым математическим критериям, наилучшим образом соответствует данным.

Регрессионный анализ очень тесно связан с корреляционным анализом. В корреляционном анализе исследуется направление и теснота связи между количественными переменными. В регрессионном анализе исследуется форма зависимости между количественными переменными.

Регрессионный анализ используется для прогноза, анализа временных рядов, тестирования гипотез и выявления скрытых взаимосвязей в данных.

Регрессионная модель

Регрессионный анализ осуществляет поиск функции регрессионной зависимости $f(x) = E(y|x)$. Регрессия может быть представлена в виде суммы неслучайной и случайной составляющих:

$$y = f(x) + \varepsilon$$

где f — функция регрессионной зависимости, а ε — аддитивная случайная величина с нулевым матожиданием. Обычно предполагается, что величина ε имеет гауссово распределение с нулевым средним и дисперсией σ_ε^2 .

Задача нахождения регрессионной модели нескольких свободных переменных:

Задана выборка: множество $\{x_1, x_2, \dots, x_N | x \in \mathbb{R}^M\}$ значений свободных переменных и множество $\{y_1, y_2, \dots, y_N | y \in \mathbb{R}\}$ соответствующих им значений зависимой переменной. Совместно эти множества обозначаются как D - множество исходных данных $\{(x, y)_i\}$.

Регрессионная модель — параметрическое семейство функций $f(w, x)$ зависящая от параметров $w \in \mathbb{R}$ и свободных переменных x . Требуется найти наиболее вероятные параметры \bar{w} :

$$\bar{w} = \underset{w \in \mathbb{R}^W}{\operatorname{argmax}} p(y|x, w, f) = p(D|w, f)$$

Функция вероятности p зависит от гипотезы порождения данных и задается Байесовским выводом или методом наибольшего правдоподобия.

Модель линейной регрессии

Линейная регрессия предполагает, что функция f зависит от параметров модели w линейно. При этом линейная зависимость от свободной переменной x необязательна:

$$y = f(w, x) + \varepsilon = \sum_{j=0}^N w_j \cdot g_j(x) + \varepsilon.$$

В случае, когда функция $g(x) = x$ линейная регрессия имеет вид:

$$f(w, x) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_N x_N \text{ и } y = \sum_{j=0}^N w_j \cdot x_j + \varepsilon,$$

где x_j — компоненты вектора x (регрессоры или факторы модели), N — количество факторов модели, w_j — параметры модели (коэффициенты регрессии), ε — случайная ошибка модели.

Коэффициенты линейной регрессии показывают скорость изменения зависимой переменной по данному фактору, при фиксированных остальных факторах (в линейной модели эта скорость постоянна): $\forall j \quad w_j = \frac{\partial f}{\partial x_j} = \text{const}$

Параметр w_0 , при котором нет факторов, называют часто константой. Формально — это значение функции при нулевом значении всех факторов. Для аналитических целей удобно считать, что константа — это параметр при «факторе», равном 1, т.е. $x_0 = 1$.

Нелинейная регрессия

Частный случай регрессионного анализа, в котором рассматриваемая регрессионная модель является функцией, зависящей от параметров и свободных переменных. Главное, что зависимость от параметров предполагается нелинейной.

Задана выборка из M пар (x_i, y_i) и регрессионная модель $f(w, x)$, которая зависит от параметров $w = (w_1, \dots, w_W)$ и свободной переменной x . Требуется найти такие значения параметров, которые доставляли бы минимум сумме квадратов регрессионных остатков SSE (RSS):

$$S(w) = \sum_{i=1}^M (y_i - f(w, x_i))^2 = \sum_{i=1}^M r_i^2,$$

где $r_i = y_i - f(w, x_i)$ для $i = 1, \dots, M$.

Для нахождения минимума функции S , приравняем к нулю её первые частные производные параметрам w :

$$\frac{\partial S}{\partial w_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial w_j} = 0, \text{ где } j = 1, \dots, n.$$

Так как функция S в общем случае может не иметь единственного минимума, то сначала назначается начальное значение вектора параметров w_0 и далее приближаться к оптимальному вектору по шагам:

$$w_j \approx w_j^{k+1} = w_j^k + \Delta w_j$$

где k — номер итерации, Δw_j — вектор шага.

Для нахождения оптимальных параметров нелинейных регрессионных моделей используются метод сопряжённых градиентов, метод Ньютона-Гаусса или алгоритм Левенберга-Марквардта.

17. Регрессионный анализ. Оценки качества регрессии.

Набор методов моделирования измеряемых данных и исследования их свойств, относится к разделам математической статистики и машинного обучения. Осуществляет исследование влияния одной или нескольких независимых переменных X_1, X_2, \dots, X_p на зависимую переменную Y .

Независимые переменные называют регрессорами, предикторами или объясняющими переменными, а зависимая переменная является результирующей, критериальной или регрессантом. Терминология *зависимых и независимых переменных* отражает лишь математическую зависимость переменных, а не причинно-следственные отношения.

Наиболее распространенный вид регрессионного анализа — линейная регрессия, когда находят линейную функцию, которая, согласно определенным математическим критериям, наилучшим образом соответствует данным.

Регрессионный анализ очень тесно связан с корреляционным анализом. В корреляционном анализе исследуется направление и теснота связи между количественными переменными. В регрессионном анализе исследуется форма зависимости между количественными переменными.

Регрессионный анализ используется для прогноза, анализа временных рядов, тестирования гипотез и выявления скрытых взаимосвязей в данных.

Оценки качества регрессии (MSE, RMSE, MAE)

Средняя квадратичная ошибка (Mean Squared Error, MSE)

MSE применяется в ситуациях, когда нам надо подчеркнуть большие ошибки и выбрать модель, которая дает меньше больших ошибок прогноза (ошибки прогноза возводятся в квадрат).

$$MSE = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

Корень из средней квадратичной ошибки (Root Mean Squared Error, RMSE)

RMSE получается из MSE путем извлечения корня.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2}$$

Средняя абсолютная ошибка (англ. Mean Absolute Error, MAE)

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

Среднеабсолютный функционал слабее штрафует за большие отклонения по сравнению со среднеквадратичным, и поэтому менее чувствителен к выбросам. При использовании любого из этих функционалов может быть полезно проанализировать, какие объекты вносят наибольший вклад в общую ошибку.

Коэффициент детерминации R -квадрат

Коэффициент детерминации измеряет долю дисперсии, объясненную моделью, в общей дисперсии целевой переменной. Фактически, данная мера качества — это нормированная среднеквадратичная ошибка.

$$R^2 = 1 - \frac{\sum_{i=1}^n (f(x_i) - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

Коэффициент детерминации принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость. При оценке регрессионных моделей это интерпретируется как соответствие модели данным.

Для приемлемых моделей $R^2 > 50\%$. Модели с $R^2 > 80\%$ можно признать достаточно хорошими. Значение $R^2 = 1$ означает функциональную зависимость между переменными.

Оценки качества регрессии (MAPE, SMAPE, MASE)

Средняя абсолютная процентная ошибка (Mean Absolute Percentage Error, MAPE)

$$MAPE = 100\% \times \frac{1}{n} \sum_{i=1}^n \frac{|y_i - f(x_i)|}{|y_i|}$$

Это коэффициент, не имеющий размерности, с простой интерпретацией. Его можно измерять в долях или процентах. Если MAPE=11.4%, то это говорит о том, что ошибка составила 11,4% от фактических значений.

Симметричная MAPE (Symmetric MAPE, SMAPE)

$$SMAPE = \frac{1}{n} \sum_{i=1}^n \frac{2|y_i - f(x_i)|}{|y_i| + |f(x_i)|}$$

Средняя абсолютная масштабированная ошибка (Mean absolute scaled error, MASE)

$$MASE = \frac{T-1}{h} \frac{\sum_{j=1}^h |e_{T+j}|}{\sum_{t=2}^T |Y_t - Y_{t-1}|}$$

MASE имеет дело с двумя суммами: числитель соответствует тестовой выборке, знаменатель — обучающей. Ошибка не зависит от масштабов данных и является симметричной: то есть положительные и отрицательные отклонения от факта рассматриваются в равной степени.

Проблема MASE в том, что её тяжело интерпретировать. Например, MASE=1.21 ни о чём, по сути, не говорит. Это просто означает, что ошибка прогноза оказалась в 1.21 раза выше среднего абсолютного отклонения ряда в первых разностях, и ничего более.

18. Обучение с учителем (supervised learning). Постановка задачи классификации, методы классификации, основные метрики качества классификации.

Обучение с учителем (supervised learning)

- Постановка задачи регрессии, линейная регрессия, логистическая, основные метрики качества регрессии.
- Постановка задачи классификации, методы классификации, основные метрики качества классификации.

Регрессионная модель

Регрессионный анализ осуществляет поиск функции регрессионной зависимости $f(x) = E(y|x)$. Регрессия может быть представлена в виде суммы неслучайной и случайной составляющих:

$$y = f(x) + \varepsilon$$

где f — функция регрессионной зависимости, а ε — аддитивная случайная величина с нулевым матожиданием. Обычно предполагается, что величина ε имеет гауссово распределение с нулевым средним и дисперсией σ_ε^2 .

Задача нахождения регрессионной модели нескольких свободных переменных:

Задана выборка: множество $\{x_1, x_2, \dots, x_N | x \in \mathbb{R}^M\}$ значений свободных переменных и множество $\{y_1, y_2, \dots, y_N | y \in \mathbb{R}\}$ соответствующих им значений зависимой переменной. Совместно эти множества обозначаются как D - множество исходных данных $\{(x, y)_i\}$.

Регрессионная модель — параметрическое семейство функций $f(w, x)$ зависящая от параметров $w \in \mathbb{R}$ и свободных переменных x . Требуется найти наиболее вероятные параметры \bar{w} :

$$\bar{w} = \underset{w \in \mathbb{R}^W}{\operatorname{argmax}} p(y|x, w, f) = p(D|w, f)$$

Функция вероятности p зависит от гипотезы порождения данных и задается Байесовским выводом или методом наибольшего правдоподобия.

Классификация

Классификация (classification) — это задача присвоения меток класса (class label) наблюдениям (Observation) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь **класс** — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.

Методы:

1. Наивный байесовский классификатор;
2. Логистическая регрессия;
3. Линейный дискриминант Фишера;
4. Нейросетевой подход;
5. Деревья решений.

...

Метрики качества классификации

Матрица ошибок (Confusion Matrix)

Основной инструмент для сравнения моделей, показывает, сколько объектов каждого класса было верно и неверно классифицировано, позволяет определить, в какой степени алгоритм делает ошибки при классификации каждого класса.

В случае бинарной кластеризации, матрица ошибок состоит из четырех основных элементов:

- True Positives (TP): Количество объектов, которые были правильно классифицированы как положительные (верное предсказание положительного класса).
- False Positives (FP): Количество объектов, которые были неправильно классифицированы как положительные (ложное предсказание положительного класса), а ошибка классификации называется ошибкой I рода (**ложная тревога**).
- True Negatives (TN): Количество объектов, которые были правильно классифицированы как отрицательные (верное предсказание отрицательного класса).
- False Negatives (FN): Количество объектов, которые были неправильно классифицированы как отрицательные (ложное предсказание отрицательного класса), а ошибка классификации называется ошибкой II рода (**пропуск цели**).

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	TN	FN

Точность (Precision): показывает, как много из объектов, предсказанных как положительные, действительно принадлежат положительному классу.

$$Precision = \frac{TP}{TP + FP}$$

Полнота (Recall): показывает, как много из всех фактически положительных объектов было правильно обнаружено моделью.

$$Recall = \frac{TP}{TP + FN}$$

Специфичность (Specificity): показывает, как много из объектов отрицательного класса было правильно классифицировано.

$$Specificity = \frac{TN}{TN + FP}$$

F-мера (F1-score): гармоническое среднее точности и полноты. Учитывает оба аспекта, что делает ее полезной в случаях, когда оба показателя важны.

$$F1\text{-score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Среднегармоническая F-мера (F_β): Для нивелирования перекоса в балансе классов используется коэффициент β .

$$F_\beta = (1 + \beta) * \frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

Точность точности (accuracy): сравнение по доле правильно классифицированных экземпляров из общего числа экземпляров в выборке.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Метрики качества классификации ROC-AUC

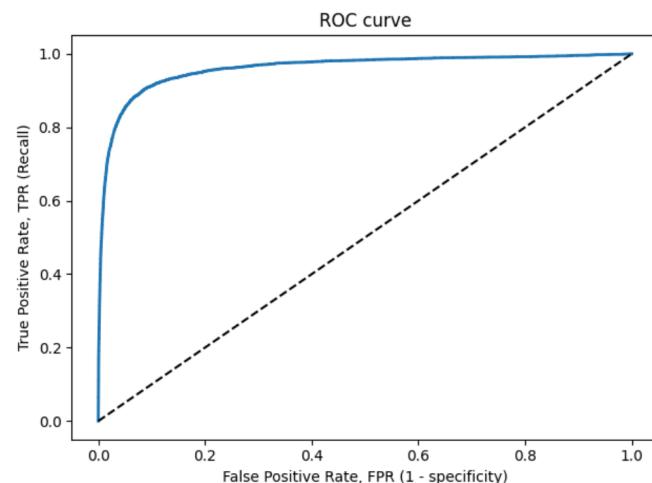
ROC — Receiver Operating Characteristic curve / AUC — Area Under Curve

ROC-кривая показывает зависимость между долей ложноположительных и долей истинно положительных классификаций. Данная оценка работы алгоритма классификации рассчитывается как площадь под ROC кривой, которая строится в координатах TPR (True Positive Rate) и FPR (False Positive Rate).

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

Алгоритм построения кривой:

1. Запустить классификатор на тестовой выборке;
2. Отсортировать результаты по уверенности классификатора в принадлежности объекта к классу;
3. Пока не кончились элементы:
 - Взять объект с максимальной уверенностью;
 - Сравнить метку с реальной;
 - Пересчитать TPR и FPR на взятых объектах;
 - Поставить точку;
4. Построить кривую по точкам.

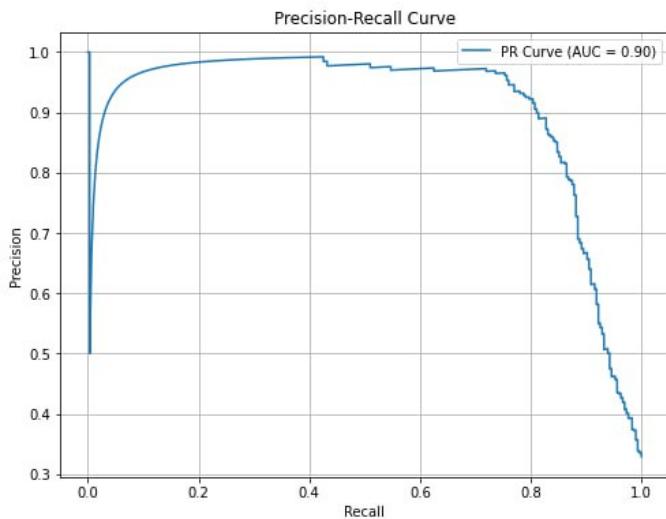


PR-Curve (Precision-Recall Curve) — это график, который отображает зависимость точности (Precision) от полноты (Recall) для различных пороговых значений классификации.

Как построить PR-curve?

1. Сортируем объекты
Расставляем объекты в порядке убывания вероятности класса "1", предсказанной моделью.
2. Начинаем с нуля
Считаем, что все объекты — это класс "0", и рассчитываем Precision и Recall.
3. Добавляем объекты по очереди
Берем первый объект (с самой высокой вероятностью класса 1) и переносим его в класс "1". Считаем новые значения рассчитываем Precision и Recall.
Повторяем это для следующего объекта, и так далее, пока все не окажутся в классе "1".
4. Строим кривую
Каждый раз, когда добавляем объект, записываем пару значений Precision и Recall и отмечаем её на графике.

Если кривая ближе к верхнему правому углу, это означает, что модель работает хорошо: она правильно определяет положительные классы, почти не ошибаясь. Если модель путает классы (например, часто называет класс "0" классом "1" и наоборот), то кривая будет ближе к левому нижнему углу. Для случайных предсказаний форма кривой будет зависеть от баланса классов в выборке. Чтобы на основании кривой получить числовую метрику, рассчитывают площадь под PR-кривой (**Precision-Recall AUC**).



19. Классификация - Байесовский подход. Цепочка расчетов и формула Байеса

Классификация

Классификация (*classification*) — это задача присвоения меток класса (*class label*) наблюдениям (*Observation*) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь **класс** — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.

Байесовский классификатор

Широкий класс алгоритмов классификации, основанный на принципе максимума апостериорной вероятности. Для классифицируемого объекта вычисляются функции правдоподобия каждого из классов, по ним вычисляются апостериорные вероятности классов. Байесовский классификатор использует оценку апостериорного максимума (*Maximum a posteriori estimation*) для определения наиболее вероятного класса. Объект относится к тому классу, для которого апостериорная вероятность максимальна.

Байесовский подход к классификации основан на теореме, утверждающей, что если плотности распределения каждого из классов известны, то искомый алгоритм можно выписать в явном аналитическом виде. Более того, этот алгоритм оптимальен, то есть обладает минимальной вероятностью ошибок.

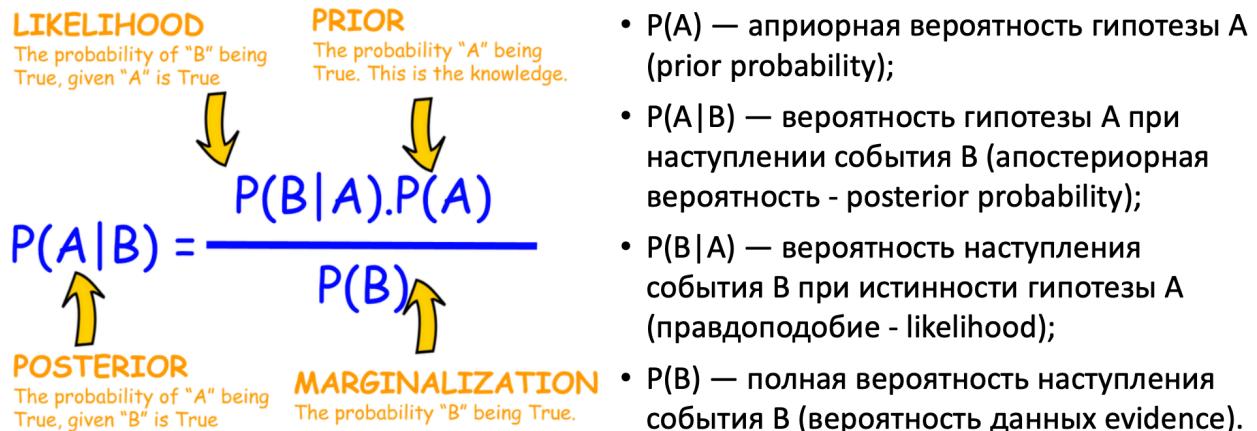
На практике плотности распределения классов не известны. Их приходится оценивать (восстанавливать) по обучающей выборке. В результате байесовский алгоритм перестаёт быть оптимальным, так как восстановить плотность по выборке можно только с некоторой погрешностью. Чем короче выборка, тем выше шансы подогнать распределение под конкретные данные и столкнуться с эффектом переобучения.

К числу байесовских методов классификации относятся:

Наивный байесовский классификатор	Метод парзеновского окна
Линейный дискриминант Фишера	Метод радиальных базисных функций (RBF)
Квадратичный дискриминант	Логистическая регрессия

Теорема Байеса

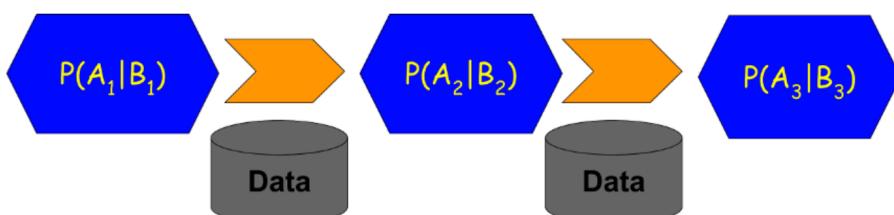
Теорема Байеса (или формула Байеса) — одна из основных теорем элементарной теории вероятностей, которая позволяет определить вероятность какого-либо события при условии, что произошло другое статистически взаимозависимое с ним событие. Другими словами, по формуле Байеса можно более точно пересчитать вероятность, взяв в расчёт как ранее известную информацию, так и данные новых наблюдений.



Цепочка расчетов и формула Байеса

Лучшее в байесовском выводе - это возможность использовать предшествующие знания в форме априорного вероятностного члена в числителе теоремы Байеса.

- Предварительные знания - это не что иное, как вычисленная вероятность теста, которая затем возвращается к следующему тесту.
- Для случаев, когда уровень распространенности среди населения в целом чрезвычайно низок, один из способов повысить уверенность в результате теста - назначить последующий тест, если первый результат теста окажется положительным.
- Апостериорная вероятность первого теста становится априорной вероятностью для второго теста, т.е. $P(U^+)$ для второго теста уже не общий показатель распространенности, а вероятность из первого теста.
- Расчетная (апостериорная) вероятность первого теста 8,9%, во втором teste она возрастает до 65,4%, а третий положительный тест дает значение 97,3%.
- Следовательно, неточный тест можно использовать несколько раз, чтобы обновить мнение с помощью последовательного применения правила Байеса.



Chaining of Bayes' rule for updating probabilities as the data comes in

20. Модель наивного байесовского классификатора

Модель наивного байесовского классификатора

Вероятностная модель для классификатора — это условная модель:

$$p(C|F_1, \dots, F_n)$$

над зависимой переменной класса C с малым количеством результатов или классов, зависящая от нескольких переменных F_1, \dots, F_n .

Используя теорему Байеса, запишем:

$$p(C | F_1, \dots, F_n) = \frac{p(C) p(F_1, \dots, F_n | C)}{p(F_1, \dots, F_n)}.$$

На практике интересен лишь числитель этой дроби, так как знаменатель не зависит от C и значения свойств F_i даны, так что знаменатель — константа.

Числитель эквивалентен совместной вероятности модели $p(C, F_1, \dots, F_n)$, которая может быть переписана, используя повторные приложения определений условной вероятности:

$$\begin{aligned} p(C, F_1, \dots, F_n) &= p(C)p(F_1, \dots, F_n | C) \\ &= p(C)p(F_1 | C)p(F_2, \dots, F_n | C, F_1) \\ &= p(C)p(F_1 | C)p(F_2 | C, F_1)p(F_3, \dots, F_n | C, F_1, F_2) \\ &= p(C)p(F_1 | C)p(F_2 | C, F_1) \dots p(F_n | C, F_1, F_2, F_3, \dots, F_{n-1}) \end{aligned}$$

«Наивные» предположения условной независимости

Предположим, что каждое свойство F_i условно независимо от любого другого свойства F_j при $j \neq i$. Это означает, что если вероятность появления события $F_i | C$, не зависит от F_j , значит F_j можно отбросить

$$p(F_i | C, F_j) = p(F_i | C)$$

таким образом, совместная модель может быть выражена как:

$$p(C, F_1, \dots, F_n) = p(C) \cdot p(F_1 | C) \cdot p(F_2 | C) \cdot p(F_3 | C) \cdots p(F_n | C) = p(C) \prod_{i=1}^n p(F_i | C)$$

Это означает, что из предположения о независимости, условное распределение по классовой переменной C может быть выражено так:

$$p(C | F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^n p(F_i | C)$$

где $Z = p(F_1, \dots, F_n)$ — это масштабный множитель, зависящий только от F_1, \dots, F_n , то есть константа, если значения переменных известны.

Оценка параметров

Все параметры модели могут быть аппроксимированы относительными частотами из набора данных обучения. Это оценки максимального правдоподобия вероятностей. Непрерывные свойства, как правило, оцениваются через нормальное распределение. В качестве математического ожидания и дисперсии вычисляются статистики — среднее арифметическое и среднеквадратическое отклонение соответственно.

Если данный класс и значение свойства никогда не встречаются вместе в наборе обучения, тогда оценка, основанная на вероятностях, будет равна нулю. Это проблема, так как при перемножении нулевая оценка приведет к потере информации о других вероятностях. Поэтому предпочтительно проводить небольшие поправки во все оценки вероятностей так, чтобы никакая вероятность не была строго равна нулю.

Построение классификатора по вероятностной модели

Наивный байесовский классификатор объединяет модель с правилом решения. Одно общее правило должно выбрать наиболее вероятную гипотезу; оно известно как *апостериорное правило принятия решения* (*maximum a posteriori* - MAP). Соответствующий классификатор — это функция *classify*, определённая следующим образом:

$$\text{Classify } (f_1, \dots, f_n) = \arg \max_c \left(p(C = c) \cdot \prod_{i=1}^n p(F_i = f_i | C = c) \right)$$

Пример: байесовская фильтрация спама

Байесовская фильтрация спама — метод для фильтрации спама, основанный на применении наивного байесовского классификатора, опирающегося на прямое использование теоремы Байеса.

При обучении фильтра для каждого встреченного в письмах слова высчитывается и сохраняется его «вес» — оценка вероятности того, что письмо с этим словом — спам. В простейшем случае в качестве оценки используется частота: «появлений в спаме / появлениях всего».

При проверке вновь пришедшего письма вероятность «спамовости» вычисляется по формуле (*Classify*) для множества гипотез.

$$P(B) = \sum_{i=1}^N P(A_i)P(B|A_i)$$

В данном случае «гипотезы» — это слова, и для каждого слова «достоверность гипотезы» $P(A_i) = N_{wordi}/N_{words\ total}$ — доля этого слова в письме, а «зависимость события от гипотезы» $P(B|A_i)$ — вычисленный ранее «вес» слова. То есть «вес» письма в данном случае — усреднённый «вес» всех его слов.

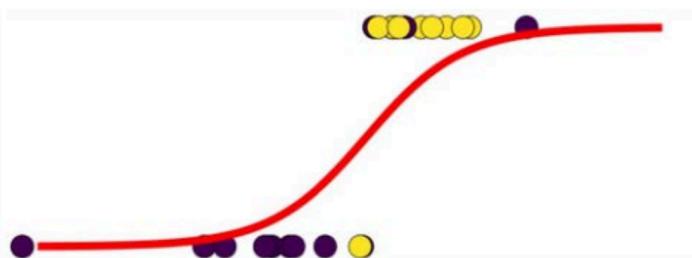
Отнесение письма к «спаму» или «не-спаму» производится по тому, превышает ли его «вес» некую планку, заданную пользователем (обычно берут 60-80 %). После принятия решения по письму в базе данных обновляются «веса» для вошедших в него слов.

21. Логистическая регрессия. Подбор параметров: метод Ньютона.

Логистическая регрессия или **логит-модель** (*logit model*) — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

Применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков.

- вводится так называемая зависимая переменная $y \in \{0, 1\}$ (0 - событие не произошло и 1 - событие произошло);
- множество независимых переменных (также называемых признаками, предикторами или регрессорами) — $x_1, x_2, \dots, x_n \in \mathbb{R}$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной;
- для простоты записи вводится фиктивный признак $x_0 = 1$.



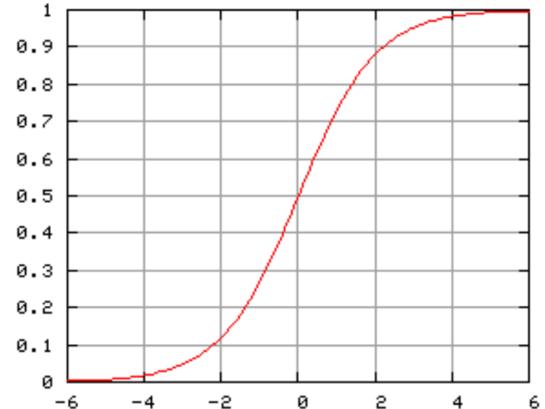
Делается предположение о том, что вероятность наступления события $y = 1$ равна:

$$P\{y = 1|x\} = f(z), \quad z = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

где

- x – вектор-столбец значений независимых переменных $1, x_1, x_2, \dots, x_n$
- θ – вектор-столбец параметров (коэффициентов регрессии) – вещественные числа $\theta_0, \dots, \theta_n$
- $f(z)$ – логистическая функция (сигмоида или логит-функция):

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



Т.к. y принимает значения 0 и 1, то вероятность принять значение 0 равна:

$$P\{y = 0|x\} = 1 - f(z) = 1 - f(\theta^T x)$$

Функция распределения y при заданном x :

$$P\{y|x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \quad y \in \{0, 1\}$$

Логистическая регрессия: подбор параметров

Для подбора параметров $\theta_0, \dots, \theta_n$ необходимо составить обучающую выборку, состоящую из наборов значений независимых переменных и соответствующих им значений зависимой переменной y .

- Формально, это множество пар $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$, где $x^{(i)} \in \mathbb{R}^n$ и $y^{(i)} \in \{0, 1\}$.
- Обычно используется метод максимального правдоподобия, согласно которому подбираются параметры θ , максимизирующие значение функции правдоподобия на обучающей выборке:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} L(\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^m P\{y = y^{(i)} | x = x^{(i)}\}$$

Определяются оценки максимального правдоподобия (maximum likelihood estimates), для которых значения параметров являются наиболее «правдоподобными» по отношению к наблюдаемым данным.

Подбор параметров: метод Ньютона

Максимизация функции правдоподобия эквивалентна максимизации её логарифма:

$$\begin{aligned}\ln L(\theta) &= \sum_{i=1}^m \ln P\{y = y^{(i)} | x = x^{(i)}\} = \sum_{i \in I_1} \ln P_i(\theta) + \sum_{i \in I_0} \ln(1 - P_i(\theta)) \\ &= \sum_{i=1}^m [y^{(i)} \ln f(\theta^T x^{(i)}) + (1 - y^{(i)}) \ln(1 - f(\theta^T x^{(i)}))]\end{aligned}$$

где $\theta^T x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_n x_n^{(i)}$, $f(z) = \frac{1}{1+e^{-z}}$ I_0, I_1 - множества наблюдений, для которых $y^{(i)} = 0$ и $y^{(i)} = 1$ соответственно.

Можно показать, что градиент: $grad = \sum_i (y^{(i)} - f(\theta^T x)) x^{(i)}$

Гессиан H функции правдоподобия равен: $H = -\sum_i P_i(1 - P_i) X_i^T X_i \leq 0$

Гессиан функции — симметрическая квадратичная форма, описывающая поведение функции во втором порядке. Для функции $f(\theta)$ дважды дифференцируемой в точке θ .

Гессиан всюду отрицательно определенный, поэтому логарифмическая функция правдоподобия всюду вогнута. Для поиска максимума можно использовать метод Ньютона, который в этом случае будет всегда сходиться

$$\theta_{t+1} = \theta_t - (H(\theta_t))^{-1} \cdot grad_t(\theta_t) = \theta_t - \Delta\theta_t$$

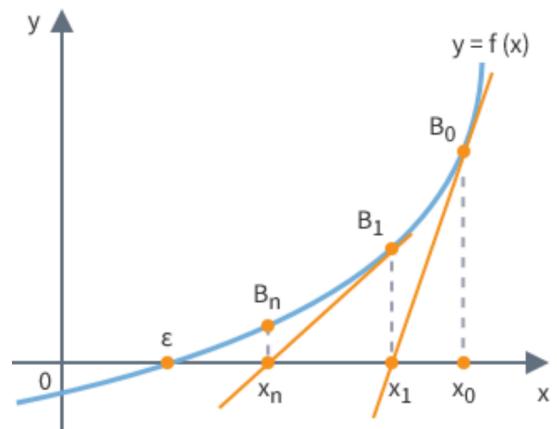
Метод Ньютона (метод касательных)

Итерационный численный метод нахождения корня (нуля) или поиска экстремума заданной функции многих переменных. Поиск решения осуществляется путём построения последовательных приближений.

Основная идея: задаётся начальное приближение вблизи предположительного корня, строится касательная к графику исследуемой функции в точке приближения, для которой находится пересечение с осью абсцисс. Эта точка берётся в качестве следующего приближения и т.д.

Алгоритм нахождения численного решения уравнения $f(x) = 0$ сводится к итерационной процедуре вычисления:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

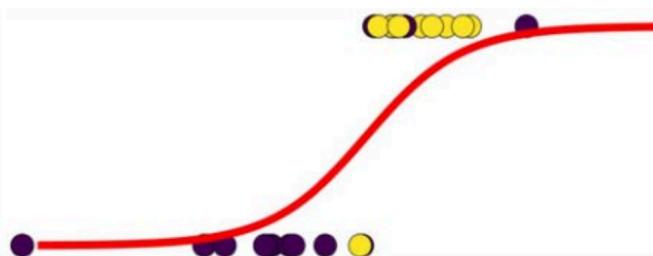


22. Логистическая регрессия. Метод градиентного спуска.

Логистическая регрессия или **логит-модель** (*logit model*) — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

Применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков.

- вводится так называемая зависимая переменная $y \in \{0, 1\}$ (0 - событие не произошло и 1 - событие произошло);
- множество независимых переменных (также называемых признаками, предикторами или регрессорами) — $x_1, x_2, \dots, x_n \in \mathbb{R}$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной;
- для простоты записи вводится фиктивный признак $x_0 = 1$.



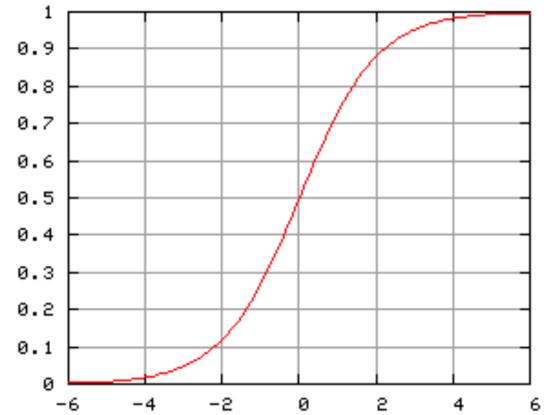
Делается предположение о том, что вероятность наступления события $y = 1$ равна:

$$P\{y = 1|x\} = f(z), \quad z = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

где

- x – вектор-столбец значений независимых переменных $1, x_1, x_2, \dots, x_n$
- θ – вектор-столбец параметров (коэффициентов регрессии) – вещественные числа $\theta_0, \dots, \theta_n$
- $f(z)$ – логистическая функция (сигмоида или логит-функция):

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



Т.к. y принимает значения 0 и 1, то вероятность принять значение 0 равна:

$$P\{y = 0|x\} = 1 - f(z) = 1 - f(\theta^T x)$$

Функция распределения y при заданном x :

$$P\{y|x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \quad y \in \{0, 1\}$$

Логистическая регрессия: подбор параметров

Для подбора параметров $\theta_0, \dots, \theta_n$ необходимо составить обучающую выборку, состоящую из наборов значений независимых переменных и соответствующих им значений зависимой переменной y .

- Формально, это множество пар $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$, где $x^{(i)} \in \mathbb{R}^n$ и $y^{(i)} \in \{0, 1\}$.
- Обычно используется метод максимального правдоподобия, согласно которому подбираются параметры θ , максимизирующие значение функции правдоподобия на обучающей выборке:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} L(\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^m P\{y = y^{(i)} | x = x^{(i)}\}$$

Определяются оценки максимального правдоподобия (maximum likelihood estimates), для которых значения параметров являются наиболее «правдоподобными» по отношению к наблюдаемым данным.

Подбор параметров: градиентный спуск

Максимизация функции правдоподобия эквивалентна максимизации её логарифма:

$$\begin{aligned} \ln L(\theta) &= \sum_{i=1}^m \log P\{y = y^{(i)} | x = x^{(i)}\} \\ &= \sum_{i=1}^m [y^{(i)} \ln f(\theta^T x^{(i)}) + (1 - y^{(i)}) \ln(1 - f(\theta^T x^{(i)}))], \end{aligned}$$

где $\theta^T x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_n x_n^{(i)}$, $f(z) = \frac{1}{1+e^{-z}}$,

а градиент $\nabla = \sum_i (y^{(i)} - f(\theta^T x)) x^{(i)}$.

Для максимизации этой функции может быть применён, например, метод градиентного спуска. Он заключается в выполнении следующих итераций, начиная с некоторого начального значения параметров θ :

$$\theta_{t+1} = \theta_t + \lambda \nabla \ln L(\theta) = \theta_t + \lambda \sum_{i=1}^m (y^{(i)} - f(\theta^T x^{(i)})) x^{(i)}, \lambda > 0.$$

Метод градиентного спуска

Численный метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента.

Пусть целевая функция имеет вид: $F(\vec{x}): \mathbb{X} \rightarrow \mathbb{R}$.

Задача оптимизации сформулирована следующим образом: $F(\vec{x}) \rightarrow \min_{\vec{x} \in \mathbb{X}}$ (найти минимум).

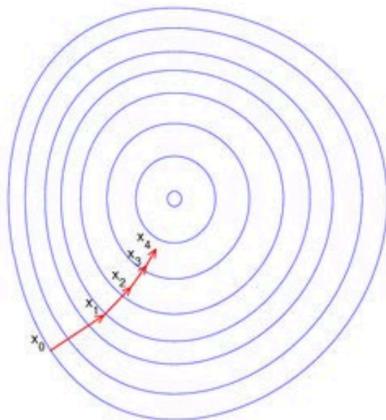
Основная идея метода заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla F$:

$$\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}),$$

где

- градиентом $\text{grad } F = \nabla F$ называется n -мерный вектор $\left(\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \dots, \frac{\partial F}{\partial x_n} \right)$, компоненты которого равны частным производным F по всем ее аргументам.
- $\lambda^{[j]}$ задает скорость градиентного спуска и может быть выбрана:
 - постоянной (в этом случае метод может не сходиться);
 - убывающей в процессе градиентного спуска;
 - гарантирующей наискорейший спуск, тогда для поиска минимума $F(\vec{x})$ получаем:

$$\lambda^{[j]} = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j+1]}) = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}))$$

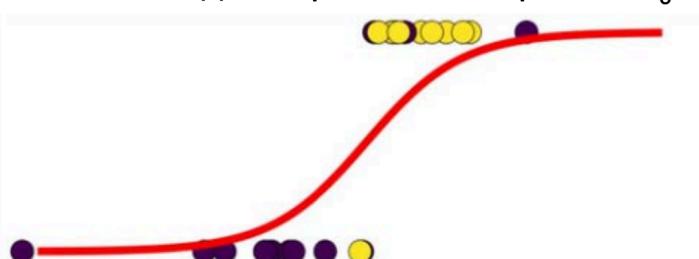


23. Логистическая регрессия и представление в виде однослойной нейронной сети

Логистическая регрессия или **логит-модель** (*logit model*) — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

Применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков.

- вводится так называемая зависимая переменная $y \in \{0, 1\}$ (0 - событие не произошло и 1 - событие произошло);
- множество независимых переменных (также называемых признаками, предикторами или регрессорами) — $x_1, x_2, \dots, x_n \in \mathbb{R}$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной;
- для простоты записи вводится фиктивный признак $x_0 = 1$.



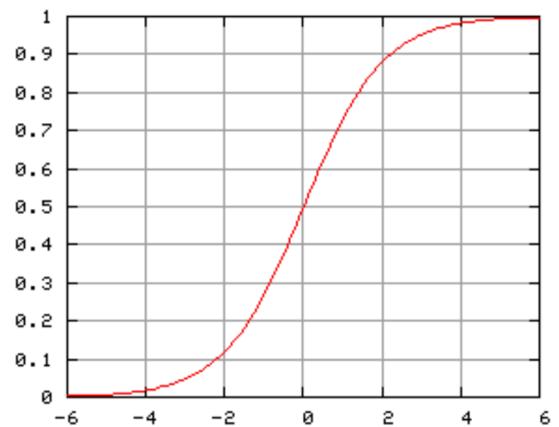
Делается предположение о том, что вероятность наступления события $y = 1$ равна:

$$P\{y = 1|x\} = f(z), \quad z = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

где

- x – вектор-столбец значений независимых переменных $1, x_1, x_2, \dots, x_n$
- θ – вектор-столбец параметров (коэффициентов регрессии) – вещественные числа $\theta_0, \dots, \theta_n$
- $f(z)$ – логистическая функция (сигмоида или логит-функция):

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



Т.к. y принимает значения 0 и 1, то вероятность принять значение 0 равна:

$$P\{y = 0|x\} = 1 - f(z) = 1 - f(\theta^T x)$$

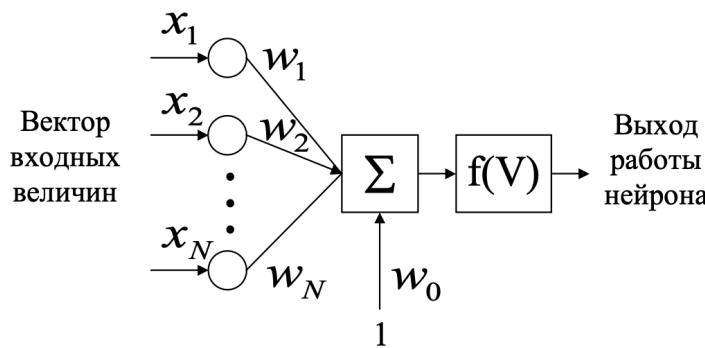
Функция распределения y при заданном x :

$$P\{y|x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \quad y \in \{0, 1\}$$

Представление в виде однослойной нейронной сети

Логистическую регрессию можно представить в виде однослойной нейронной сети с сигмоидальной функцией активации, веса которой есть коэффициенты логистической регрессии, а вес поляризации — константа регрессионного уравнения.

Структурная схема нейрона:



$$V = \sum_{i=0}^N w_i \cdot x_i$$

$$f(V) = \frac{1}{1 + \exp(-bV)}$$

Однослойная нейронная сеть может успешно решить лишь задачу линейной сепарации. Поэтому возможности по моделированию нелинейных зависимостей у логистической регрессии отсутствуют.

24. Линейный дискриминантный анализ и линейный дискриминант Фишера.

Классификация

Классификация (classification) — это задача присвоения меток класса (class label) наблюдениям (Observation) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь **класс** — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.

- В линейном случае мы хотим спроектировать точки в размерность 1 (на нормаль разделяющей гиперплоскости) так, чтобы в этой одномерной размерности они хорошо разделялись
- В этом смысле классификация — это метод радикального сокращения размерности.
- Рассмотрим классификацию с этих позиций и в каком-то смысле попробуем добиться оптимальности.

Для решения задач классификации с двумя или более классами большинство алгоритмов машинного обучения применяют какое-то преобразование к входным данным с эффектом уменьшения исходных входных измерений до меньшего числа. Цель состоит в том, чтобы спроектировать данные в новое пространство. Затем, после проектирования, алгоритм пытается классифицировать точки путем нахождения линейного разделения.

Линейный дискриминант Фишера

Линейный дискриминантный анализ (ЛДА), а также связанный с ним линейный дискриминант Фишера — методы статистики и машинного обучения, применяемые для нахождения линейных комбинаций признаков, наилучшим образом разделяющих два или более класса объектов или событий. Полученная комбинация может быть использована в качестве линейного классификатора или для сокращения размерности пространства признаков перед последующей классификацией.

Линейный дискриминант Фишера в первоначальном значении - метод, определяющий расстояние между распределениями двух разных классов объектов или событий. Он может использоваться в задачах машинного обучения при статистическом (байесовском) подходе к решению задач классификации.

Предположим, что обучающая выборка удовлетворяет помимо базовых гипотез байесовского классификатора также следующим гипотезам:

- Классы распределены по нормальному закону
- Матрицы ковариаций классов равны

Тогда статистический подход приводит к линейному дискриминанту, и именно этот алгоритм классификации в настоящее время часто понимается под термином **линейный дискриминант Фишера**

Первая идея

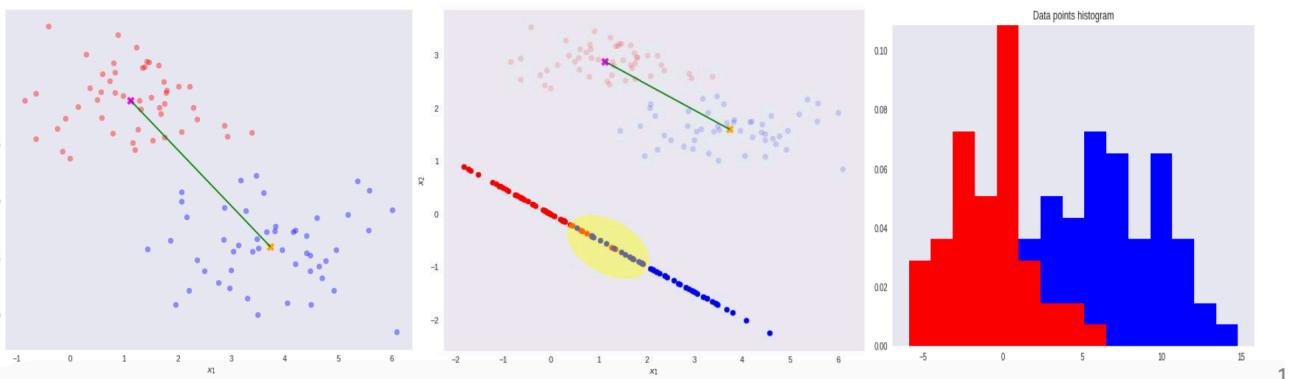
Рассмотрим два класса C_1 и C_2 с N_1 и N_2 точками.

Первая идея – найти серединный перпендикуляр между центрами кластеров. Вычислим средние векторы m_1 и m_2 .

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n, \quad m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

Другими словами, хотим проецировать данные на вектор W , соединяющий центры кластеров.

Важно отметить, что любой вид проекции на меньшее измерение может повлечь некоторую потерю информации. В этом сценарии обратите внимание, что эти два класса четко разделимы (линейей) в их исходном пространстве. После проецирования данные демонстрируют некоторое перекрытие классов - это показано желтым эллипсом на графике и гистограммой справа.



Идея, предложенная Фишером

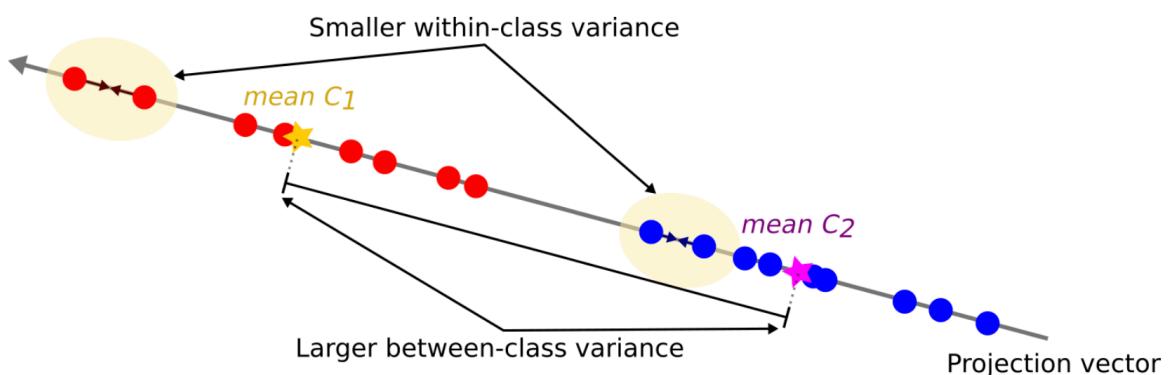
Максимизировать функцию, которая обеспечит наибольшее разделение между спроектированными центрами классов, а также даст наименьшую дисперсию внутри каждого класса, тем самым минимизируя перекрытие классов.

Другими словами, линейный дискриминант Фишера выбирает проекцию, которая максимизирует разделение классов. Для этого он максимизирует соотношение между дисперсией между классами и дисперсией внутри класса.

Для проецирования данных в меньшее измерение и во избежание перекрытия классов метод поддерживает 2 свойства.

- Большая разница между классами наборов данных.
- Небольшая дисперсия в каждом из классов наборов данных.

Обратите внимание, что большая разница между классами означает, что прогнозируемые средние классы должны быть как можно дальше друг от друга. Напротив, небольшая внутрикласовая дисперсия позволяет удерживать проецируемые точки данных ближе друг к другу в рамках одного класса.



Поиск проекции с указанными свойствами

Выборочная дисперсия в проекции $y_n = W^T x_n$

$$S_1^2 = \sum_{n \in C_1} (y_n - m_1)^2, S_2^2 = \sum_{n \in C_2} (y_n - m_2)^2$$

Чтобы найти проекцию с указанными свойствами, Линейный дискриминант Фишера вычисляет весовой вектор W по следующему критерию (критерий Фишера):

$$J(W) = \frac{(m_2 - m_1)^2}{S_1^2 + S_2^2} = \frac{W^T S_B W}{W^T S_W W}$$

Числитель – between-class variance (межклассовая дисперсия).

Знаменатель - within-class variance (внутриклассовая дисперсия).

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$
$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

После дифференцирования по W получим, что $J(W)$ максимален при:

$$(W^T S_B W) S_W W = (W^T S_W W) S_B W$$

Причем $S_B W$ будет в направлении $m_2 - m_1$, а длина W нас не интересует.

Получаем:

$$W \propto S_W^{-1} (m_2 - m_1)$$

То есть, W (наше желаемое преобразование) прямо пропорционально обратной внутриклассовой ковариации. Матрица умножает на разность классов.

25. Нейросетевой подход к решению задач классификации.

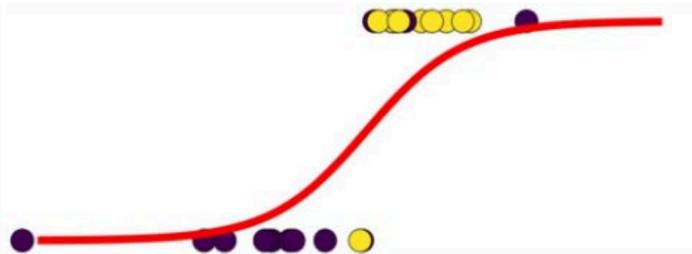
Классификация

Классификация (classification) — это задача присвоения меток класса (class label) наблюдениям (Observation) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь **класс** — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.

Логистическая регрессия или **логит-модель** (*logit model*) — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

Применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков.

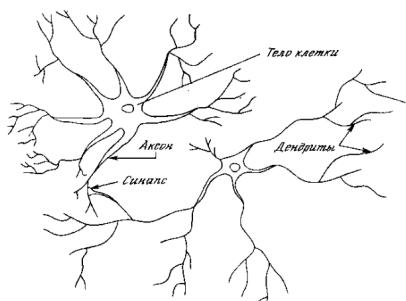
- вводится так называемая зависимая переменная $y \in \{0, 1\}$ (0 - событие не произошло и 1 - событие произошло);
- множество независимых переменных (также называемых признаками, предикторами или регрессорами) — $x_1, x_2, \dots, x_n \in \mathbb{R}$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной;
- для простоты записи вводится фиктивный признак $x_0 = 1$.



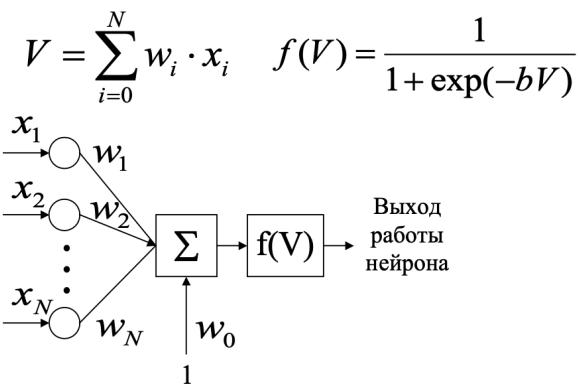
Логистическая регрессия и математическая модель нейрона

Логистическую регрессию можно представить в виде однослоевой нейронной сети с сигмоидальной функцией активации, веса которой есть коэффициенты логистической регрессии.

Структурная схема нейрона:



Вектор входных величин



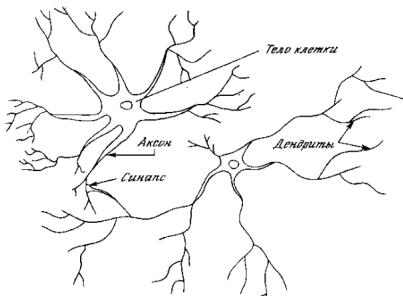
Первой попыткой математически описать процесс функционирования нейрона следует считать работу Мак-Каллока и Питтса, написанную ими еще в 1943 году. Отдельный нейрон состоит из "тела" (по аналогии с биологическим нейроном), суммирующего сигналы, поступающие с синапсов. При этом каждый синапс может передавать либо возбуждающий, либо тормозящий сигнал. В зависимости от соотношения возбуждающих и тормозящих сигналов нейрон либо возбуждается и передает дальше возбуждающий сигнал, либо тормозится и передает тормозящий сигнал.

Следующий шаг в описании процесса функционирования нейронных сетей внес Розенблатт, описавший функционирование персептрона (1957). Персептрон Розенблатта представляет полноценную математическую модель отдельного нейрона.

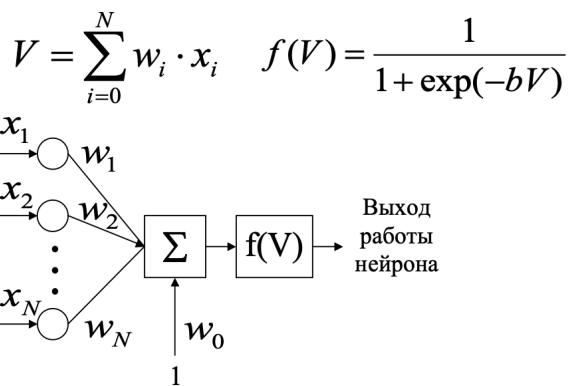
26. Принципы построения искусственных нейронных

сетей. Нейрон, функция активации, обучение.

Структурная схема нейрона:



Вектор
входных
величин



Первой попыткой математически описать процесс функционирования нейрона следует считать работу Мак-Каллока и Питтса, написанную ими еще в 1943 году. Отдельный нейрон состоит из "тела" (по аналогии с биологическим нейроном), суммирующего сигналы, поступающие с синапсов. При этом каждый синапс может передавать либо возбуждающий, либо тормозящий сигнал. В зависимости от соотношения возбуждающих и тормозящих сигналов нейрон либо возбуждается и передает дальше возбуждающий сигнал, либо тормозится и передает тормозящий сигнал.

Следующий шаг в описании процесса функционирования нейронных сетей внес Розенблattt, описавший функционирование персептрона (1957). Персептрон Розенблattt представляет полноценную математическую модель отдельного нейрона.

Искусственная нейронная сеть (Мак-Каллок и Питтс)

Мак-Каллок и Питтс исходили из предположений о том, что нервная система биологических существ состоит из большого количества нейронов, каждый из которых имеет несколько входных и один выходной сигнал. Описывая процесс функционирования нейронной сети, они приняли следующие допущения (1943 год):

- активность нейрона удовлетворяет принципу "все или ничего";
- возбуждению нейрона предшествует период накопления возбуждений фиксированного количества его синапсов, причем это число не зависит от предыдущей активности и расположения синапсов в нейроне;
- единственным запаздыванием в нервной системе, имеющим значение, является синаптическая задержка;
- активность какого-либо тормозящего синапса исключает возбуждение нейрона в рассматриваемый момент времени;
- с течением времени структура нейронной сети не меняется.

Поскольку нейрон имеет как входные, так и выходные сигналы, то различные нейроны могут быть соединены между собой, образуя нейронную сеть.

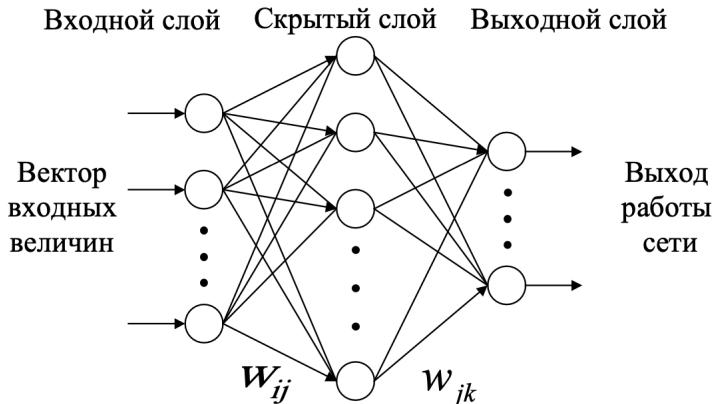
Искусственная нейронная сеть Artificial neural network (ANN)

Математическая модель, а также ее программные или аппаратные реализации, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма.

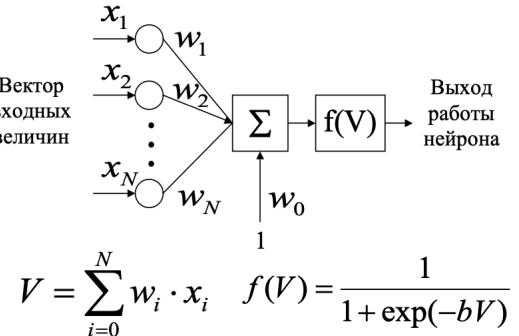
Под искусственной нейронной сетью в дальнейшем будем понимать сеть искусственных нейронов, соединенных между собой. Здесь предполагается, что нейроны могут соединяться между собой произвольным образом и образовывать таким образом разнообразные нейронные структуры.

Искусственная нейронная сеть

Многослойный персептрон



Структурная схема нейрона:



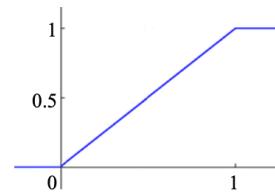
Функционирование: $Y_k(x) = Y_k(x_1, \dots, x_l) = f\left(\sum_{j=0}^m w_{jk} f\left(\sum_{i=0}^n w_{ij} x_i\right)\right)$

Сигналы проходят от входного слоя нейронов через скрытые слои к выходным элементам. В процессе функционирования каждый узел многослойной сети проектирует свой входной вектор на вектор весов посредством скалярного произведения (т.е. вычисляет взвешенную сумму входного вектора). После этого результаты проекции подвергаются нелинейным преобразованиям. Их цель - усилить те характеристики, за которые отвечает соответствующий узел.

Функции активации

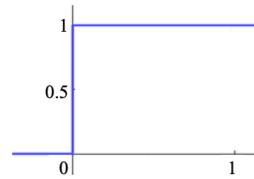
- Линейная: $f(V) = V$

- Линейная с насыщением: $f(V) = \begin{cases} 0, & V \leq 0 \\ 1, & V \geq 1 \\ V, & 0 < V < 1 \end{cases}$



- Ступенчатая (пороговая):

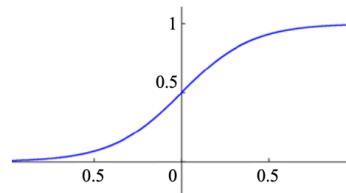
$$f(V) = \begin{cases} 1, & V \geq 0 \\ 0, & V < 0 \end{cases}$$



- Многопороговая

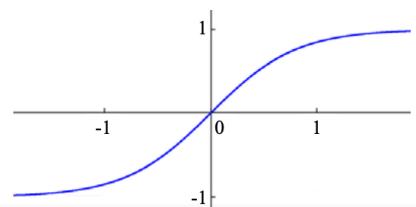
- Сигмоидная (логистическая):

$$f(V) = \frac{1}{1 + \exp(-bV)}$$



- Гиперболический тангенс :

$$f(V) = \tanh(V) \equiv \frac{\exp(bV) - 1}{\exp(bV) + 1}$$



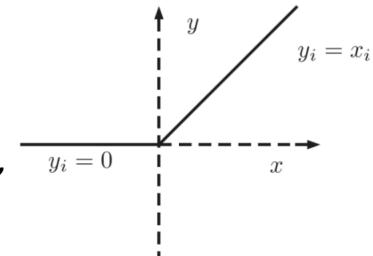
Семейство функций активации ReLU

- **ReLU (Rectified linear unit)**

$$f(x) = \max(0, x)$$

✓ Отсутствие эффекта насыщения;

✓ «*Dying ReLU Problem*» (проблема «умирающего» ReLU), до 40% ReLU нейронов никогда не активируются.

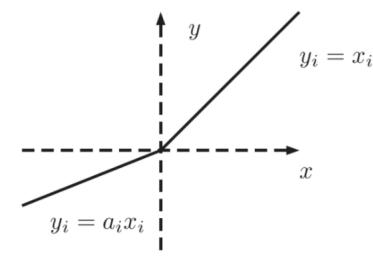


- **Leaky ReLU** - ReLU с «утечкой»

для $x < 0$ имеет небольшое отрицательное значение, определяемое малым угловым коэффициентом:

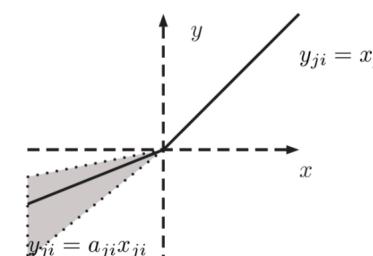
$$f(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

где α – малая константа порядка 0,01.



- **Parametric ReLU**

(Parameteric rectified linear unit, PReLU)



- **Randomized ReLU**

(Randomized leaky rectified linear unit)

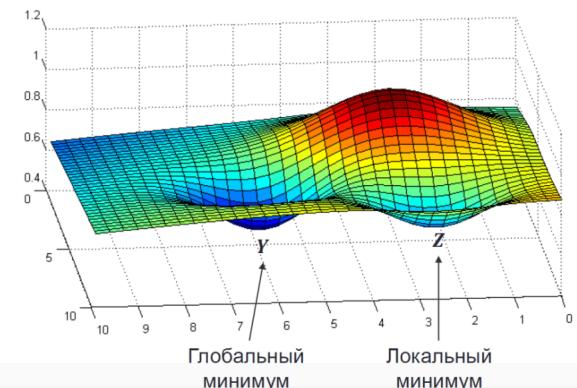
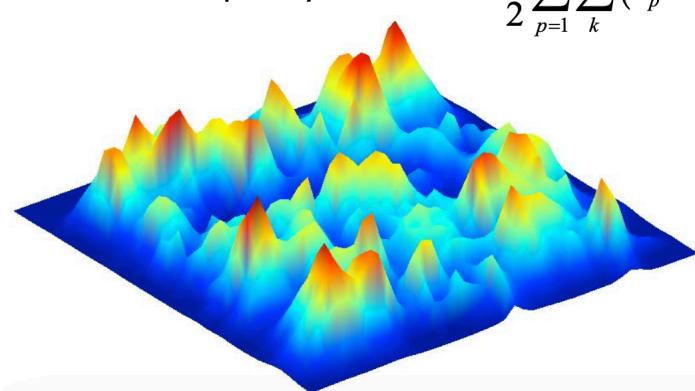
Многослойный персепtron (методы обучения)

Обучение нейронной сети - интерактивный процесс корректировки синаптических весов и порогов. В процессе обучения нейронная сеть получает и обобщает знания об окружающей среде и тех данных, с которыми ей придется оперировать. Существуют два концептуальных подхода к обучению нейронных сетей: обучение с учителем и обучение без учителя.

Обучение персептрана:

- Обучение Хебба (без учителя);
- Стохастические методы обучения (“имитация отжига”);
- Обратное распространение ошибки (Backpropagation).

Ошибка сети при обучении: $E = \frac{1}{2} \sum_{p=1}^P \sum_k (u_p^k - y_p^k)^2$



Обучение Хебба

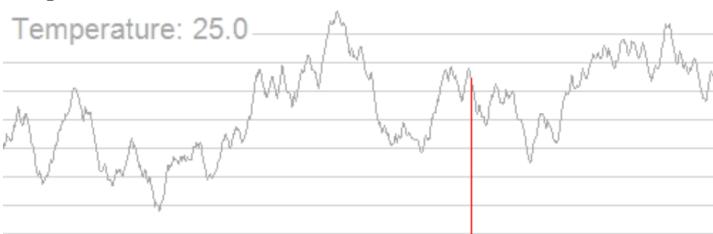
Предложена модель обучения без учителя, в которой синаптическая сила (вес) возрастает, если активированы оба нейрона, источник и приемник. Таким образом, часто используемые пути в сети усиливаются и феномен привычки и обучения через повторение получает объяснение.

$$w_{ij}(n+1) = w_{ij}(n) + \eta * OUT_i * OUT_j$$

$w_{ij}(n)$ – значение веса от нейрона i к нейрону j до подстройки, $w_{ij}(n+1)$ – значение веса от нейрона i к нейрону j после подстройки, η – коэффициент скорости обучения, OUT_i – выход нейрона i и вход нейрона j , OUT_j – выход нейрона j .

Стохастические методы обучения «имитация отжига»

Выполняются псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям.



Для обучения сети может быть использована следующая процедура:

1. Выбрать значения весов случайным образом. Предъявить множество входов и вычислить получающиеся выходы.
2. Сравнить полученные значения с желаемыми выходами и вычислить величину разности между ними (сумма квадратов разностей). Целью обучения является минимизация этой разности (часто называют **целевой функцией**).
3. Выбрать случайный весовой коэффициент и подкорректировать его на небольшое случайное значение. Если коррекция помогает (уменьшает целевую функцию), то сохранить ее, в противном случае вернуться к первоначальному значению веса. Случайное значение корректировки постепенно уменьшается со временем.
4. Повторять шаг 3 до тех пор, пока сеть не будет обучена в достаточной степени.

Скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени, чтобы была достигнута сходимость к глобальному минимуму.

$$T(t) = \frac{T_0}{\log(1+t)}$$

T(t) – искусственная температура как функция времени;
 T₀ – начальная искусственная температура;
 t – искусственное время.

Обратное распространение ошибки (Backpropagation)

Алгоритм обучения с учителем. Фактически является алгоритмом градиентного спуска, минимизирующим суммарную квадратичную ошибку:

$$E = \frac{1}{2} \sum_{k=1}^P \sum_i (d_k^i - y_k^i)^2$$

где d_k^i - желаемый выход нейрона i ,
 y_k^i - текущий выход нейрона i последнего слоя.

Синаптические веса настраиваются в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad \text{изменение веса: } \Delta w_{ij} = -\varepsilon \frac{\partial E_k}{\partial w_{ij}} = -\varepsilon \delta_k^j x_k^i$$

где ε - длина шага в направлении, обратном к градиенту;

δ_k^j - вычисляется через аналогичные множители из последующего слоя, и ошибка передается в обратном направлении.

Для выходных элементов: $\delta_k^j = -(d_k^j - y_k^j) f'(V_k^j) = -(d_k^j - y_k^j) \cdot y_k^j \cdot (1 - y_k^j)$

Для скрытых элементов: $\delta_k^j = -f'(V_k^j) \sum_h w_{jh} \delta_h^k = out_k^j \cdot (1 - out_k^j) \cdot \sum_h w_{jh} \delta_h^k$

где индекс h пробегает номера всех нейронов, на которые воздействует j -й нейрон, out_k^j - выход нейрона j , k - номер обучающего примера.

Этапы Backpropagation

Весь алгоритм разбивается на следующие этапы:

1. подать на вход сети один из требуемых образов и определить значения выходов нейронов сети .
2. рассчитать δ_k^j для выходного слоя сети по формуле

$$\delta_k^j = -(d_k^j - y_k^j) f'(V_k^j) = -(d_k^j - y_k^j) \cdot y_k^j \cdot (1 - y_k^j)$$

и рассчитать изменения весов Δw_{ij} выходного слоя.

3. Рассчитать по формуле
4. Скорректировать все веса сети. Если ошибка существенна, то перейти на шаг 1.

$$\delta_k^j = -f'(V_k^j) \sum_h w_{jh} \delta_h^k = out_k^j \cdot (1 - out_k^j) \cdot \sum_h w_{jh} \delta_h^k$$

значения δ_k^j и Δw_{ij} для остальных слоев нейросети, двигаясь от выходного слоя к входному.

27. Многослойный персептрон. Выбор числа слоев, числа нейронов в скрытом слое. Методы обучения.

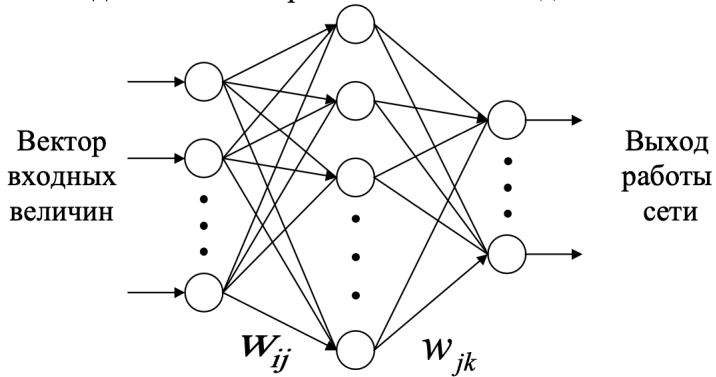
Искусственная нейронная сеть Artificial neural network (ANN)

Математическая модель, а также ее программные или аппаратные реализации, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма.

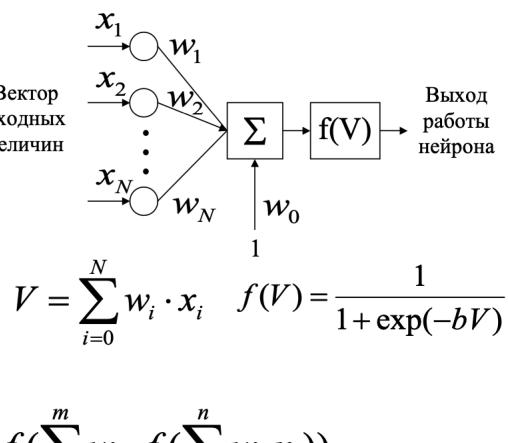
Под искусственной нейронной сетью в дальнейшем будем понимать сеть искусственных нейронов, соединенных между собой. Здесь предполагается, что нейроны могут соединяться между собой произвольным образом и образовывать таким образом разнообразные нейронные структуры.

Искусственная нейронная сеть Многослойный персептрон

Входной слой Скрытый слой Выходной слой Структурная схема нейрона:



Структурная схема нейрона:

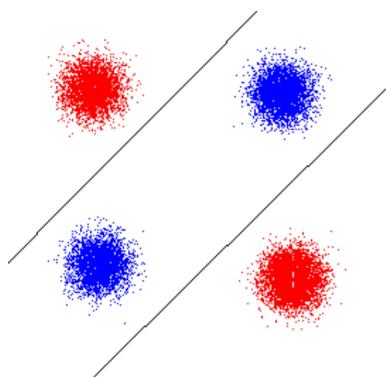


Функционирование: $Y_k(x) = Y_k(x_1, \dots, x_l) = f\left(\sum_{j=0}^m w_{jk} f\left(\sum_{i=0}^n w_{ij} x_i\right)\right)$

Сигналы проходят от входного слоя нейронов через скрытые слои к выходным элементам. В процессе функционирования каждый узел многослойной сети проектирует свой входной вектор на вектор весов посредством скалярного произведения (т.е. вычисляет взвешенную сумму входного вектора). После этого результаты проекции подвергаются нелинейным преобразованиям. Их цель - усилить те характеристики, за которые отвечает соответствующий узел.

Многослойный персептрон – число слоев

Марвин Минский и Сеймур Паперт в своей работе "Персептроны" доказали, что простейшие однослойные нейронные сети, состоящие из входного и выходного слоев (известные, как линейный персептрон), способны решать только линейно разделимые задачи и обеспечивают универсальную линейную аппроксимацию. Это ограничение преодолимо путем добавления скрытых слоев и использования многослойных нейронных сетей.



В общем виде можно сказать, что при решении задач классификации в сети с одним скрытым слоем, входной вектор преобразуется в некоторое новое пространство, возможно с другой размерностью, а затем поверхности, соответствующие нейронам выходного слоя, разделяют его на классы. Таким образом, сеть распознает не только характеристики исходных данных, но и "характеристики характеристик", сформированные скрытым слоем.

Нейронные сети с числом скрытых слоев большим единицы называются глубокими (deep neural network). Они могут содержать меньшее число нейронов в каждом слое, чем сети с одним скрытым слоем, реализующие то же самое отображение. Однако строгой методики сопоставления таких сетей пока не существует.

Многослойный персептрон число нейронов в каждом слое

Количество нейронов входного слоя напрямую зависит от размерности исходного пространства входных данных (пространства признаков) и от методов их кодирования.

Количество нейронов скрытого слоя

Проблема выбора количества скрытых элементов многослойного персептрона заключается в том, что с одной стороны, число скрытых элементов должно быть достаточным для решения поставленной задачи, а с другой не должно быть слишком большим, чтобы обеспечить необходимую обобщающую способность сети и избежать переобучения. То есть, нельзя просто выбрать теоретический максимум числа весовых коэффициентов, так как в этом случае сеть научится иметь дело только с теми данными, которые предъявлялись в процессе тренировки, и, поэтому обобщающая способность сети будет слабой.

Количество нейронов выходного слоя зависит от решаемой задачи и, также как для входного слоя, от способов кодирования

Многослойный персептрон

Число нейронов скрытого слоя - оценки

Для оценки числа нейронов в скрытом слое однородной нейронной сети можно воспользоваться формулой для оценки необходимого числа синаптических весов N_w в многослойной сети с сигмоидальными функциями активации:

$$\frac{N_y N_p}{1 + \log_2(N_p)} \leq N_w \leq N_y \left(\frac{N_p}{N_x} + 1 \right) (N_x + N_y + 1) + N_y$$

где N_y - размерность выходного сигнала, N_p - число элементов обучающей выборки, N_x - размерность входного сигнала.

Оценив необходимое число весов, можно рассчитать число нейронов в скрытых слоях. Число нейронов в сети с одним скрытым слоем составит:

$$N = \frac{N_w}{N_x + N_y}$$

Многослойный персептрон

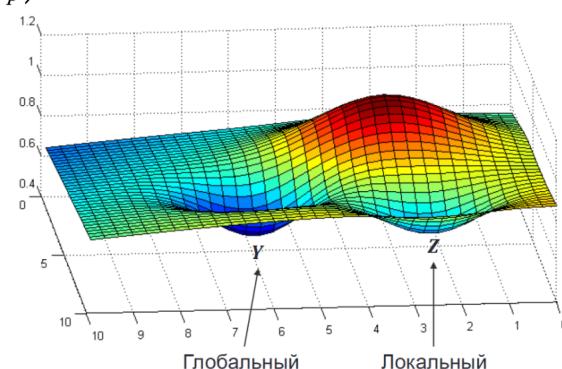
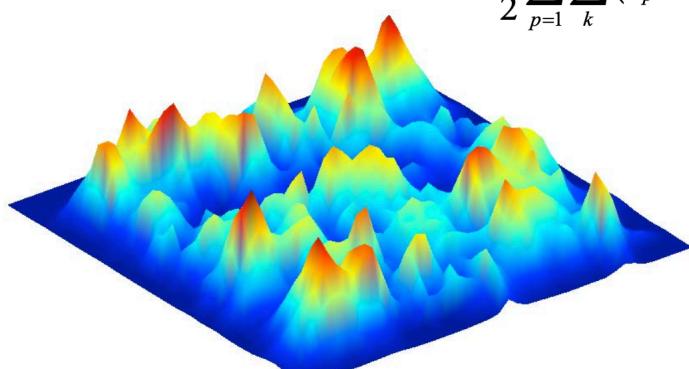
(методы обучения)

Обучение нейронной сети - интерактивный процесс корректировки синаптических весов и порогов. В процессе обучения нейронная сеть получает и обобщает знания об окружающей среде и тех данных, с которыми ей придется оперировать. Существуют два концептуальных подхода к обучению нейронных сетей: обучение с учителем и обучение без учителя.

Обучение персептрона:

- Обучение Хебба (без учителя);
- Стохастические методы обучения ("имитация отжига");
- Обратное распространение ошибки (Backpropagation).

Ошибка сети при обучении: $E = \frac{1}{2} \sum_{p=1}^P \sum_k (u_p^k - y_p^k)^2$



Обучение Хебба

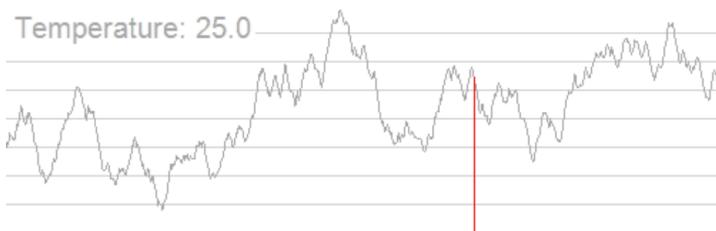
Предложена модель обучения без учителя, в которой синаптическая сила (вес) возрастает, если активированы оба нейрона, источник и приемник. Таким образом, часто используемые пути в сети усиливаются и феномен привычки и обучения через повторение получает объяснение.

$$w_{ij}(n+1) = w_{ij}(n) + \eta * OUT_i * OUT_j$$

$w_{ij}(n)$ – значение веса от нейрона i к нейрону j до подстройки, $w_{ij}(n+1)$ – значение веса от нейрона i к нейрону j после подстройки, η – коэффициент скорости обучения, OUT_i – выход нейрона i и вход нейрона j , OUT_j – выход нейрона j .

Стохастические методы обучения «Имитация отжига»

Выполняются псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям.



Для обучения сети может быть использована следующая процедура:

1. Выбрать значения весов случайным образом. Предъявить множество входов и вычислить получающиеся выходы.
2. Сравнить полученные значения с желаемыми выходами и вычислить величину разности между ними (сумма квадратов разностей). Целью обучения является минимизация этой разности (часто называют *целевой функцией*).
3. Выбрать случайный весовой коэффициент и подкорректировать его на небольшое случайное значение. Если коррекция помогает (уменьшает целевую функцию), то сохранить ее, в противном случае вернуться к первоначальному значению веса. Случайное значение корректировки постепенно уменьшается со временем.
4. Повторять шаг 3 до тех пор, пока сеть не будет обучена в достаточной степени.

Скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени, чтобы была достигнута сходимость к глобальному минимуму.

$$T(t) = \frac{T_0}{\log(1+t)}$$

T(t) – искусственная температура как функция времени;
T₀ – начальная искусственная температура;
t – искусственное время.

Обратное распространение ошибки (Backpropagation)

Алгоритм обучения с учителем. Фактически является алгоритмом градиентного спуска, минимизирующим суммарную квадратичную ошибку:

$$E = \frac{1}{2} \sum_{k=1}^P \sum_i (d_k^i - y_k^i)^2$$

где d_k^i - желаемый выход нейрона i ,
 y_k^i - текущий выход нейрона i последнего слоя.

Синаптические веса настраиваются в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

изменение веса: $\Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}} = -\varepsilon \delta_k^j x_k^i$

где ε - длина шага в направлении, обратном к градиенту;

δ_k^i - вычисляется через аналогичные множители из последующего слоя, и ошибка передается в обратном направлении.

Для выходных элементов: $\delta_k^j = -(d_k^j - y_k^j)f'(V_k^j) = -(d_k^j - y_k^j) \cdot y_k^j \cdot (1 - y_k^j)$

Для скрытых элементов: $\delta_k^j = -f'(V_k^j) \sum_h w_{jh} \delta_k^h = out_k^j \cdot (1 - out_k^j) \cdot \sum_h w_{jh} \delta_k^h$

где индекс h пробегает номера всех нейронов, на которые воздействует j -й нейрон, out_k^j - выход нейрона j , k - номер обучающего примера.

Этапы Backpropagation

Весь алгоритм разбивается на следующие этапы:

1. подать на вход сети один из требуемых образов и определить значения выходов нейронов сети .
2. рассчитать δ_k^j для выходного слоя сети по формуле

$$\delta_k^j = -(d_k^j - y_k^j)f'(V_k^j) = -(d_k^j - y_k^j) \cdot y_k^j \cdot (1 - y_k^j)$$

и рассчитать изменения весов Δw_{ij} выходного слоя.

3. Рассчитать по формуле

$$\delta_k^j = -f'(V_k^j) \sum_h w_{jh} \delta_k^h = out_k^j \cdot (1 - out_k^j) \cdot \sum_h w_{jh} \delta_k^h$$

значения δ_k^j и Δw_{ij} для остальных слоев нейросети, двигаясь от выходного слоя к входному.

4. Скорректировать все веса сети. Если ошибка существенна, то перейти на шаг 1.

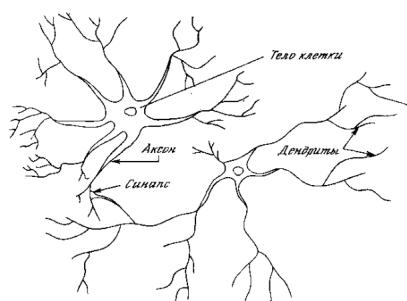
28. Нейронные сети основанные на базисных радиальных функциях.

Искусственная нейронная сеть Artificial neural network (ANN)

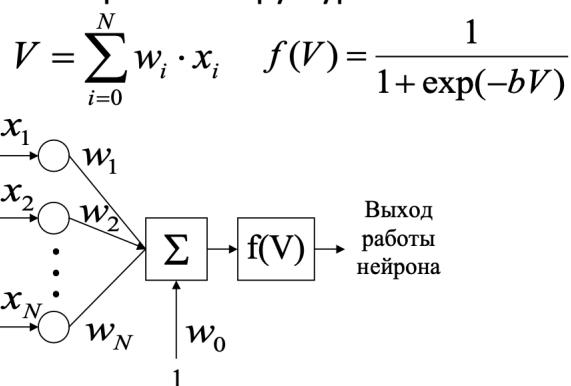
Математическая модель, а также ее программные или аппаратные реализации, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма.

Под искусственной нейронной сетью в дальнейшем будем понимать сеть искусственных нейронов, соединенных между собой. Здесь предполагается, что нейроны могут соединяться между собой произвольным образом и образовывать таким образом разнообразные нейронные структуры.

Структурная схема нейрона:

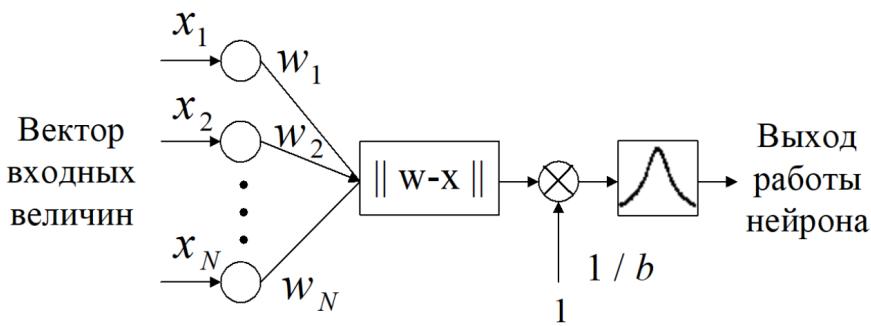


Вектор входных величин



Первой попыткой математически описать процесс функционирования нейрона следует считать работу Мак-Каллока и Питтса, написанную ими еще в 1943 году. Отдельный нейрон состоит из "тела" (по аналогии с биологическим нейроном), суммирующего сигналы, поступающие с синапсов. При этом каждый синапс может передавать либо возбуждающий, либо тормозящий сигнал. В зависимости от соотношения возбуждающих и тормозящих сигналов нейрон либо возбуждается и передает дальше возбуждающий сигнал, либо тормозится и передает тормозящий сигнал.

RBF-нейрон

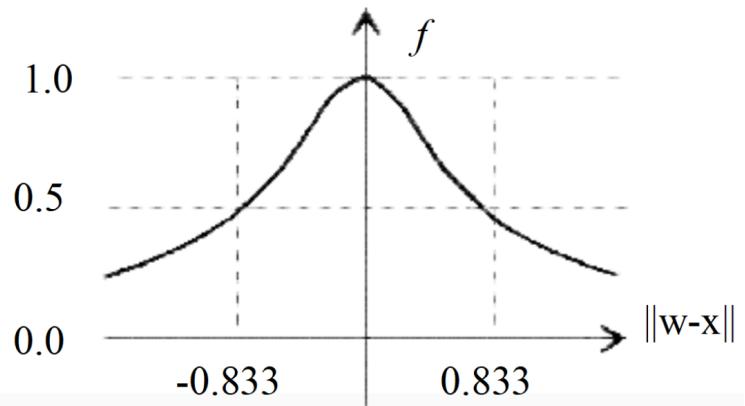


Потенциал нейрона - расстояние между векторами весовых коэффициентов и входных величин (евклидово расстояние)

$$\|w - x\| = \sqrt{\sum_{i=1}^N (w_i - x_i)^2}$$

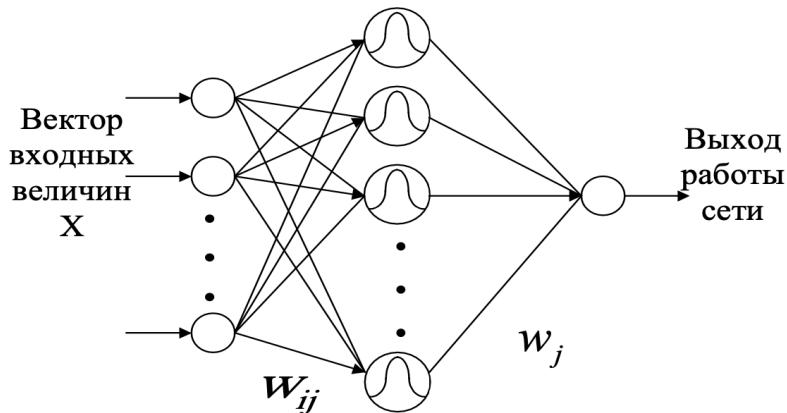
Функция активации:

$$f = e^{-\left(\frac{\|w-x\|}{b}\right)^2}$$



RBF-сеть

Входной слой Скрытый слой Выходной слой



Функционирование:

$$u_{net} = \sum_{j=1}^n w_j \cdot a_j$$

Обучение RBF-сети:

1. Формирование скрытого слоя

- Образцами обучающей выборки;
- С учетом имеющихся кластеров в данных;

2. Обучение выходного линейного слоя

- Методом псевдообратных матриц: $\bar{w} = (A^T A)^{-1} A^T \bar{u}$
- NLMS – Normalized Least Mean Squares: $w_j(t) = w_j(t-1) + \Delta(t) \cdot e_{net}(t) \cdot \frac{a_j(t)}{a^T(t)a(t)}$

29. Деревья решений. Сфера применения, решаемые задачи. Алгоритмы обучения.

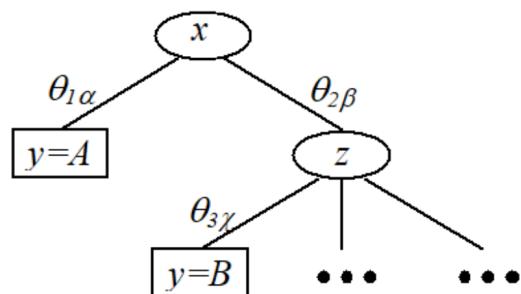
Деревья решений

Деревья решений относятся к методам поиска логических закономерностей в данных, а также являются основным подходом, применимым в теории принятия решений.

Они позволяют осуществлять решение целого класса задач классификации и регрессии в виде многошагового процесса принятия решений и используют особенности древовидных классификаторов, связанных с учетом локальных свойств классифицируемых объектов на каждом уровне и в каждом узле дерева, что позволяет реализовать как прямую, так и обратную цепочку рассуждений.

Основными достоинствами деревьев решений является:

- простота и наглядность описания процесса поиска решения;
- представление правил в виде продукции «если... то...».



Если (условие 1) \wedge (условие 2) $\wedge \dots \wedge$ (условие N) то (значение вершины вывода)

$$\theta_1(x, \alpha) \rightarrow (y = A)$$
$$\theta_2(x, \beta) \cap \theta_3(z, \gamma) \rightarrow (y = B)$$

Структура и функционирование

Дерево решений — метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.

Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — класс, для дерева регрессии — модальный интервал целевой переменной.

Чтобы попасть в лист, пример должен удовлетворять всем правилам, лежащим на пути к этому листу. Поскольку путь в дереве к каждому листу единственный, то каждый пример может попасть только в один лист, что обеспечивает единственность решения.

Задачи

Сфера применения — поддержка процессов принятия управлеченческих решений, используемая в статистике, анализе данных и машинном обучении. Задачами, решаемыми с помощью данного аппарата, являются:

- **Классификация** — отнесение объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.
- **Регрессия (численное предсказание)** — предсказание числового значения независимой переменной для заданного входного вектора.
- **Описание объектов** — набор правил в дереве решений позволяет компактно описывать объекты. Поэтому вместо сложных структур, описывающих объекты, можно хранить деревья решений.

Алгоритмы обучения деревья решений

В настоящее время разработано значительное число алгоритмов обучения деревья решений: CLS, ID3, C4.5, CART, NewId, IRule, CHAID, CN2 и т.д. Наибольшее распространение и популярность получили следующие:

- **ID3 (Iterative Dichotomizer 3)** — позволяет работать только с дискретной целевой переменной, т.е. строит только классифицирующие деревья решений.
- **C4.5** — усовершенствованная версия алгоритма ID3, в которую добавлена возможность работы с пропущенными значениями атрибутов (по версии издания Springer Science в 2008 году алгоритм занял 1-е место в топ-10 наиболее популярных алгоритмов Data Mining).
- **CART (Classification and Regression Tree)** — алгоритм обучения деревьев решений способный использовать как дискретную, так и непрерывную целевую переменную, т.е. решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

30. Деревья решений. Проблемы основных этапов построения.

Деревья решений

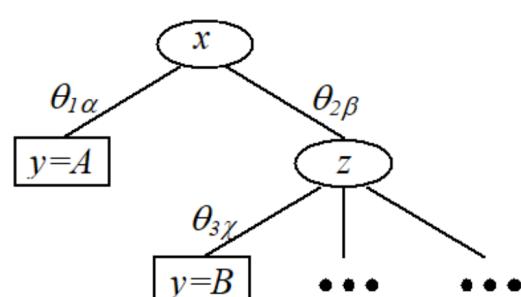
Деревья решений относятся к методам поиска логических закономерностей в данных, а также являются основным подходом, применимым в теории принятия решений.

Они позволяют осуществлять решение целого класса задач классификации и регрессии в виде многошагового процесса принятия решений и используют особенности древовидных классификаторов, связанных с учетом локальных свойств классифицируемых объектов на каждом уровне и в каждом узле дерева, что позволяет реализовать как прямую, так и обратную цепочку рассуждений.

Основными достоинствами деревьев решений является:

- простота и наглядность описания процесса поиска решения;
- представление правил в виде продукции «если... то...».

Если (условие 1) \wedge (условие 2) $\wedge \dots \wedge$ (условие N) то (значение вершины вывода)



$$\begin{aligned}\theta_1(x, \alpha) &\rightarrow (y = A) \\ \theta_2(x, \beta) \cap \theta_3(z, \chi) &\rightarrow (y = B)\end{aligned}$$

Структура и функционирование

Дерево решений — метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.

Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — класс, для дерева регрессии — модальный интервал целевой переменной.

Чтобы попасть в лист, пример должен удовлетворять всем правилам, лежащим на пути к этому листу. Поскольку путь в дереве к каждому листу единственный, то каждый пример может попасть только в один лист, что обеспечивает единственность решения.

Процесс построения

Деревья решений являются моделями, строящимися на основе обучения с учителем, следовательно, в обучающем множестве для примеров должно быть задано целевое значение.

- Целевая переменная дискретная -> дерево классификации.
- Целевая переменная непрерывная -> дерево регрессии.

Процесс построения деревьев решений заключается в рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями.

Объявление узла листом может произойти:

- содержит единственный объект, или объекты только одного класса;
- достижение некоторого условия остановки (минимально допустимое число примеров в узле или максимальная глубина дерева).

Алгоритмы построения деревьев решений относят к категории «жадных алгоритмов». Жадными называются алгоритмы, которые допускают, что локально-оптимальные решения на каждом шаге (разбиения в узлах), приводят к оптимальному итоговому решению. Поэтому на этапе построения нельзя сказать обеспечит ли выбранный атрибут, в конечном итоге, оптимальное разбиение.

Пусть задано обучающее множество S , содержащее n примеров, для каждого из которых задана метка класса $C_i (i = 1..k)$, и m атрибутов $A_j (j = 1..m)$, которые, как предполагается, определяют принадлежность объекта к тому или иному классу. Тогда возможны три случая:

- Все примеры множества S имеют одинаковую метку класса C_i (т.е. все обучающие примеры относятся только к одному классу). Дерево решений в этом случае будет представлять собой лист, ассоциированный с классом C_i .
- Множество S не содержит примеров, т.е. является пустым. Для него тоже будет создан лист. Применять правило, чтобы создать узел, к пустому множеству бессмысленно.
- Множество S содержит обучающие примеры всех классов C_k . Требуется разбить множество S на подмножества, ассоциированные с классами. Для этого:
 - выбирается один из атрибутов A_j множества S , который содержит два и более уникальных значения (a_1, a_2, \dots, a_p) , где p — число уникальных значений признака.
 - множество S разбивается на p подмножеств (S_1, S_2, \dots, S_p) , каждое из которых включает примеры, содержащие соответствующее значение атрибута.
 - выбирается следующий атрибут и разбиение повторяется.
 - процедура рекурсивно повторяется до тех пор, пока все примеры в результирующих подмножествах не окажутся одного класса.

При использовании данной методики, построение дерева решений будет происходить сверху вниз (от корневого узла к листьям).

Проблемы основных этапов построения

- Выбор атрибута, по которому будет производиться разбиение в данном узле (атрибут разбиения).
- Выбор критерия остановки обучения.
- Выбор метода отсечения ветвей (упрощения).
- Оценка точности построенного дерева.

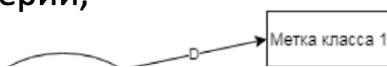
Выбор атрибута разбиения

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут.

Общее правило: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

Наиболее популярные критерии:

- Теоретико-информационный критерий;
- Статистический подход.



31. Деревья решений. Алгоритмы построения. Выбор атрибута разбиения.

Деревья решений

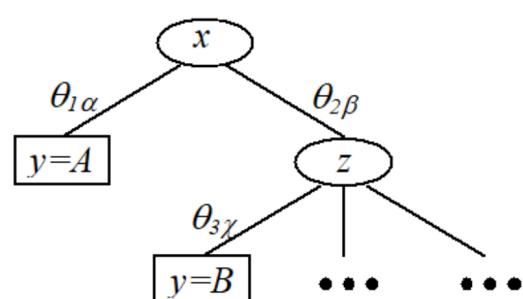
Деревья решений относятся к методам поиска логических закономерностей в данных, а также являются основным подходом, применимым в теории принятия решений.

Они позволяют осуществлять решение целого класса задач классификации и регрессии в виде многошагового процесса принятия решений и используют особенности древовидных классификаторов, связанных с учетом локальных свойств классифицируемых объектов на каждом уровне и в каждом узле дерева, что позволяет реализовать как прямую, так и обратную цепочку рассуждений.

Основными достоинствами деревьев решений является:

- простота и наглядность описания процесса поиска решения;
- представление правил в виде продукции «если... то...».

Если (условие 1) \wedge (условие 2) $\wedge \dots \wedge$ (условие N) то (значение вершины вывода)



$$\begin{aligned}\theta_1(x, \alpha) &\rightarrow (y = A) \\ \theta_2(x, \beta) \cap \theta_3(z, \chi) &\rightarrow (y = B)\end{aligned}$$

Структура и функционирование

Дерево решений — метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.

Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — класс, для дерева регрессии — модальный интервал целевой переменной.

Чтобы попасть в лист, пример должен удовлетворять всем правилам, лежащим на пути к этому листу. Поскольку путь в дереве к каждому листу единственный, то каждый пример может попасть только в один лист, что обеспечивает единственность решения.

Алгоритмы обучения дерева решений

В настоящее время разработано значительное число алгоритмов обучения дерева решений: CLS, ID3, C4.5, CART, NewId, ITrule, CHAID, CN2 и т.д. Наибольшее распространение и популярность получили следующие:

- **ID3 (Iterative Dichotomizer 3)** — позволяет работать только с дискретной целевой переменной, т.е. строит только классифицирующие деревья решений.
- **C4.5** — усовершенствованная версия алгоритма ID3, в которую добавлена возможность работы с пропущенными значениями атрибутов (по версии издания Springer Science в 2008 году алгоритм занял 1-е место в топ-10 наиболее популярных алгоритмов Data Mining).
- **CART (Classification and Regression Tree)** — алгоритм обучения деревьев решений способный использовать как дискретную, так и непрерывную целевую переменную, т.е. решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

Процесс построения

Деревья решений являются моделями, строящимися на основе обучения с учителем, следовательно, в обучающем множестве для примеров должно быть задано целевое значение.

- Целевая переменная дискретная -> дерево классификации.
- Целевая переменная непрерывная -> дерево регрессии.

Процесс построения деревьев решений заключается в рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями.

Объявление узла листом может произойти:

- содержит единственный объект, или объекты только одного класса;
- достижение некоторого условия остановки (минимально допустимое число примеров в узле или максимальная глубина дерева).

Алгоритмы построения деревьев решений относят к категории «жадных алгоритмов». Жадными называются алгоритмы, которые допускают, что локально-оптимальные решения на каждом шаге (разбиения в узлах), приводят к оптимальному итоговому решению. Поэтому на этапе построения нельзя сказать обеспечит ли выбранный атрибут, в конечном итоге, оптимальное разбиение.

Пусть задано обучающее множество S , содержащее n примеров, для каждого из которых задана метка класса $C_i (i = 1..k)$, и m атрибутов $A_j (j = 1..m)$, которые, как предполагается, определяют принадлежность объекта к тому или иному классу. Тогда возможны три случая:

- Все примеры множества S имеют одинаковую метку класса C_i (т.е. все обучающие примеры относятся только к одному классу). Дерево решений в этом случае будет представлять собой лист, ассоциированный с классом C_i .
- Множество S не содержит примеров, т.е. является пустым. Для него тоже будет создан лист. Применять правило, чтобы создать узел, к пустому множеству бессмысленно.
- Множество S содержит обучающие примеры всех классов C_k . Требуется разбить множество S на подмножества, ассоциированные с классами. Для этого:
 - выбирается один из атрибутов A_j множества S , который содержит два и более уникальных значения (a_1, a_2, \dots, a_p) , где p — число уникальных значений признака.
 - множество S разбивается на p подмножеств (S_1, S_2, \dots, S_p) , каждое из которых включает примеры, содержащие соответствующее значение атрибута.
 - выбирается следующий атрибут и разбиение повторяется.
 - процедура рекурсивно повторяется до тех пор, пока все примеры в результирующих подмножествах не окажутся одного класса.

При использовании данной методики, построение дерева решений будет происходить сверху вниз (от корневого узла к листьям).

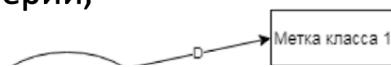
Выбор атрибута разбиения

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут.

Общее правило: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

Наиболее популярные критерии:

- Теоретико-информационный критерий;
- Статистический подход.



32. Деревья решений. Выбор атрибута разбиения. Split-Info и Gain-Ratio.

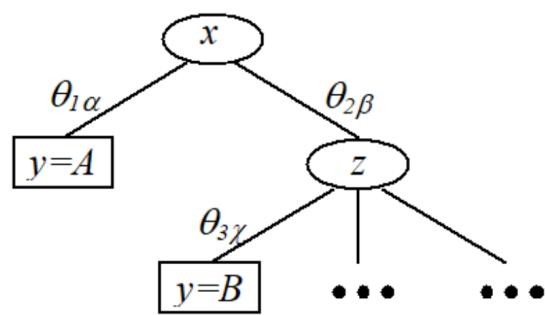
Деревья решений

Деревья решений относятся к методам поиска логических закономерностей в данных, а также являются основным подходом, применимым в теории принятия решений.

Они позволяют осуществлять решение целого класса задач классификации и регрессии в виде многошагового процесса принятия решений и используют особенности древовидных классификаторов, связанных с учетом локальных свойств классифицируемых объектов на каждом уровне и в каждом узле дерева, что позволяет реализовать как прямую, так и обратную цепочку рассуждений.

Основными достоинствами деревьев решений является:

- простота и наглядность описания процесса поиска решения;
- представление правил в виде продукции «если... то...».



Если (условие 1) \wedge (условие 2) $\wedge \dots \wedge$ (условие N) то (значение вершины вывода)

$$\theta_1(x, \alpha) \rightarrow (y = A)$$

$$\theta_2(x, \beta) \cap \theta_3(z, \chi) \rightarrow (y = B)$$

Структура и функционирование

Дерево решений — метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.

Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — класс, для дерева регрессии — модальный интервал целевой переменной.

Чтобы попасть в лист, пример должен удовлетворять всем правилам, лежащим на пути к этому листу. Поскольку путь в дереве к каждому листу единственный, то каждый пример может попасть только в один лист, что обеспечивает единственность решения.

Процесс построения

Деревья решений являются моделями, строящимися на основе обучения с учителем, следовательно, в обучающем множестве для примеров должно быть задано целевое значение.

- Целевая переменная дискретная \rightarrow дерево классификации.
- Целевая переменная непрерывная \rightarrow дерево регрессии.

Процесс построения деревьев решений заключается в рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями.

Объявление узла листом может произойти:

- содержит единственный объект, или объекты только одного класса;
- достижение некоторого условия остановки (минимально допустимое число примеров в узле или максимальная глубина дерева).

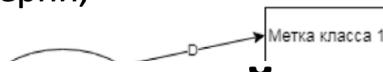
Алгоритмы построения деревьев решений относят к категории «жадных алгоритмов». Жадными называются алгоритмы, которые допускают, что локально-оптимальные решения на каждом шаге (разбиения в узлах), приводят к оптимальному итоговому решению. Поэтому на этапе построения нельзя сказать обеспечит ли выбранный атрибут, в конечном итоге, оптимальное разбиение.

Выбор атрибута разбиения

Общее правило: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

Наиболее популярные критерии:

- Теоретико-информационный критерий;
- Статистический подход.



Теоретико-информационный критерий

Критерий основан на понятиях теории информации (информационной энтропии):

$$H(S, C) = - \sum_{i=1}^n \frac{N_i}{N} \log \left(\frac{N_i}{N} \right)$$

где n — число классов в исходном подмножестве, N_i — число примеров i -го класса, N — общее число примеров в подмножестве.

Энтропия может рассматриваться как мера неоднородности подмножества по представленным в нём классам. Если классы представлены в равных долях, то неопределенность классификации и энтропия максимальны. Если все примеры в узле относятся к одному классу, т.е. $N = N_i$, логарифм от единицы обращает энтропию в ноль.

Лучшим атрибутом разбиения A_j будет тот, который обеспечит максимальное снижение энтропии результирующего подмножества относительно родительского. На практике говорят не об энтропии, а об обратной величине, которая называется информацией. Тогда лучшим атрибутом разбиения будет тот, который обеспечит максимальный прирост информации результирующего узла относительно исходного:

$$Gain(S, A) = Info(S) - Info(S_A)$$

где $Info(S)$ — информация, связанная с подмножеством S до разбиения, $Info(S_A)$ — информация, связанная с подмножеством, полученным при разбиении по атрибуту A .

$$Gain(S, A) = H(S, C) - \sum_{j=1}^p \frac{|S_j|}{|S|} H(S_j, C)$$

где S_j -множество элементов S , на которых атрибут A имеет значение a_j .

Таким образом, задача выбора атрибута разбиения в узле заключается в максимизации величины $Gain(S, A)$, называемой приростом информации (gain — прирост, увеличение).

Этот критерий впервые был применён в алгоритме ID3, а затем в C4.5 и других алгоритмах.

Проблема критерия прироста информации

Можно заметить, что критерий разбиения на основе прироста информации отдаёт предпочтение атрибутам, которые содержат большее число уникальных значений. Это связано с тем, что при этом создается большое количество узлов-потомков, дающее в итоге более равномерное распределение классов и, соответственно, меньшую энтропию разбиения. В предельном случае, если все значения атрибута будут уникальными, то для каждого примера будет создано отдельное правило и отдельный лист с единственным классом.

Как следствие, разбиение даст нулевую энтропию:

$$Info(S, A) = 0$$

и максимальный прирост информации.

Это оптимально с «точки зрения» алгоритма, но абсолютно бессмысленно с практической точки зрения, поскольку:

- значимость правил будет чрезвычайно низкая, т.к. правило действует только для конкретного объекта;
- дерево решений окажется очень сложным для понимания;
- дерево решений получится переобученным, т.е. не будет обладать обобщающей способностью.

Split-Info и Gain-Ratio

Штраф может быть представлен в виде некоторого коэффициента для значения прироста информации.

Введём в рассмотрение показатель:

$$SplitInfo(S, A) = - \sum_{j=1}^p \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}$$

который представляет собой оценку потенциальной информации, созданной при разбиении множества S на p подмножеств S_j . Тогда критерий прироста информации с учётом показателя $SplitInfo(S, A)$ будет иметь вид:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInfo(S, A)}$$

Новый критерий позволяет оценить долю информации, полученной при разбиении, которая является «полезной», то есть способствует улучшению классификации. Применение данного показателя, как правило, приводит к выбору более удачного атрибута, чем использование обычного критерия прироста информации.

Смысл этой модификации достаточно прост. $|S_j|/|S|$ — отношение числа примеров в подмножестве, полученном в результате разбиения, к числу примеров в родительском множестве $|S|$. Если в результате разбиения получается большое число подмножеств с небольшим числом примеров, что характерно для переобучения, то показатель $SplitInfo$ растет. Поскольку он стоит в знаменателе выражения, то $GainRatio$ есть обычный прирост информации, «оштрафованный» с помощью $SplitInfo$.

33. Ансамбли классификаторов. Типы Ансамблей. Теорема Кондорсе о присяжных.

Ансамбли — это контролируемые алгоритмы обучения, которые комбинируют прогнозы из двух и более алгоритмов машинного обучения для построения более точных результатов. Результаты можно комбинировать с помощью голосования или усреднения. Первое зачастую применяется в классификации, а второе — в регрессии.

Существует 3 основных типа ансамблевых алгоритмов:

- Бэггинг.** Алгоритмы обучаются и работают параллельно на разных тренировочных наборах одного размера. Затем все они тестируются на одном наборе данных, а конечный результат определяется с помощью голосования.
- Бустинг.** В этом типе алгоритмы обучаются последовательно, а конечный результат отбирается с помощью голосования с весами.
- Стекинг (наложение).** Исходя из названия, этот подход состоит из двух уровней, расположенных друг на друге. Базовый представляет собой комбинацию алгоритмов, а верхний — мета-алгоритмы, основанные на базовом уровне.

Теорема Кондорсе о присяжных

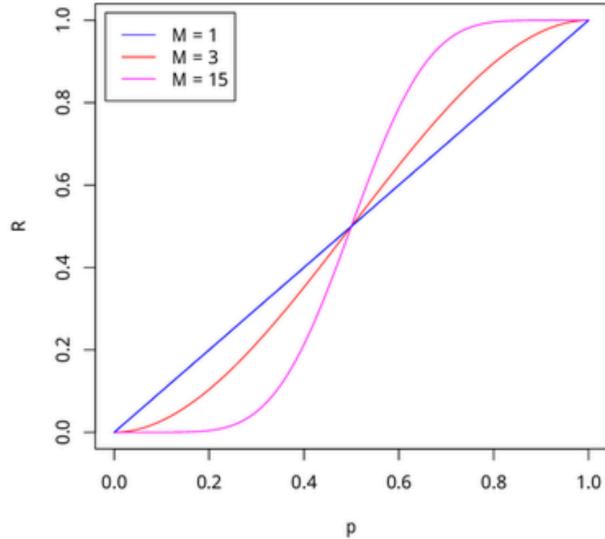
Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри, и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.

Пусть M – количество присяжных, p – вероятность правильного решения одного эксперта, R – вероятность правильного решения всего жюри, m – минимальное большинство членов жюри:

$$m = \left\lfloor \frac{M}{2} \right\rfloor + 1$$

Тогда

$$R = \sum_{i=m}^M C_M^i p^i (1-p)^{M-i}$$



34. Ансамбли классификаторов. Бэггинг. Random Forest.

Ансамбли — это контролируемые алгоритмы обучения, которые комбинируют прогнозы из двух и более алгоритмов машинного обучения для построения более точных результатов. Результаты можно комбинировать с помощью голосования или усреднения. Первое зачастую применяется в классификации, а второе — в регрессии.

Существует 3 основных типа ансамблевых алгоритмов:

- Бэггинг.** Алгоритмы обучаются и работают параллельно на разных тренировочных наборах одного размера. Затем все они тестируются на одном наборе данных, а конечный результат определяется с помощью голосования.
- Бустинг.** В этом типе алгоритмы обучаются последовательно, а конечный результат отбирается с помощью голосования с весами.
- Стекинг (наложение).** Исходя из названия, этот подход состоит из двух уровней, расположенных друг на друге. Базовый представляет собой комбинацию алгоритмов, а верхний — мета-алгоритмы, основанные на базовом уровне.

Бэггинг - Идея

Пусть обучающая выборка состоит из n объектов. Выберем из неё n примеров равновероятно, с возвращением. Получим новую выборку X^1 , в которой некоторых элементов исходной выборки не будет, а какие-то могут войти несколько раз. С помощью некоторого алгоритма b обучим на этой выборке модель $b_1(x) = b(x, X^1)$. Повторим процедуру: сформируем вторую выборку X^2 из n элементов с возвращением и с помощью того же алгоритма обучим на ней модель $b_2(x) = b(x, X^2)$. Повторив процедуру k раз, получим k моделей, обученных на k выборках. Чтобы получить одно предсказание, усредним предсказания всех моделей:

$$a(x) = \frac{1}{k} (b_1(x) + \dots + b_k(x))$$

Процесс генерации подвыборок с помощью семплирования с возвращением называется **бутстрепом (bootstrap)**, а модели $b_1(x), \dots, b_k(x)$ часто называют **базовыми алгоритмами (моделями)**. Модель $a(x)$ называется **ансамблем** этих моделей.

Бэггинг - Смещение

Будем считать, что когда мы берём матожидание по всем обучающим выборкам X , то в эти выборки включены также все подвыборки, полученные бутстрепом.

$$\begin{aligned} bias_X a(x, X) &= f(x) - \mathbb{E}_X[a(x, X)] = f(x) - \mathbb{E}_X\left[\frac{1}{k} \sum_{i=1}^k b(x, X^i)\right] = \\ &= f(x) - \frac{1}{k} \sum_{i=1}^k \mathbb{E}_X[b(x, X^i)] = f(x) - \frac{1}{k} \sum_{i=1}^k \mathbb{E}_X[b(x, X)] = \\ &= f(x) - \mathbb{E}_X[b(x, X)] = bias_X b(x, X) \end{aligned}$$

Получаем, что смещение ансамбля не изменилось по сравнению со средним смещением отдельных моделей.

Бэггинг – Разброс (Дисперсия)

$$\begin{aligned} \mathbb{V}_X[a(x, X)] &= \mathbb{E}_X[a(x, X) - \mathbb{E}_X[a(x, X)]]^2 = \\ &= \mathbb{E}_X\left[\frac{1}{k} \sum_{i=1}^k b(x, X^i) - \mathbb{E}_X\left[\frac{1}{k} \sum_{i=1}^k b(x, X^i)\right]\right]^2 = \frac{1}{k^2} \mathbb{E}_X\left[\sum_{i=1}^k (b(x, X^i) - \mathbb{E}_X[b(x, X^i)])\right]^2 = \\ &= \frac{1}{k^2} \sum_{i=1}^k \mathbb{E}_X(b(x, X^i) - \mathbb{E}_X[b(x, X^i)])^2 + \\ &\quad + \frac{1}{k^2} \sum_{k_1 \neq k_2} \mathbb{E}_X[(b(x, X^{k_1}) - \mathbb{E}_X[b(x, X^{k_1})])(b(x, X^{k_2}) - \mathbb{E}_X[b(x, X^{k_2})])] = \\ &= \frac{1}{k^2} \sum_{i=1}^k \mathbb{V}_X b(x, X^i) + \frac{1}{k^2} \sum_{k_1 \neq k_2} cov(b(x, X^{k_1}), b(x, X^{k_2})) \end{aligned}$$

Если предположить, что базовые алгоритмы некоррелированы, то:

$$\mathbb{V}_X[a(x, X)] = \frac{1}{k^2} \sum_{i=1}^k \mathbb{V}_X b(x, X^i) = \frac{1}{k^2} \sum_{i=1}^k \mathbb{V}_X b(x, X) = \frac{1}{k} \mathbb{V}_X b(x, X)$$

Дисперсия композиции в k раз меньше дисперсии отдельного алгоритма!

Random Forest (Случайный лес)

Мы делали предположение, что базовые алгоритмы некоррелированы. Однако в реальной жизни добиться этого проблематично: ведь базовые алгоритмы учат одну и ту же зависимость на пересекающихся выборках. В реальности достаточно, чтобы алгоритмы были в некоторой степени не похожи друг на друга. Развитие идеи бэггинга для решающих деревьев — **случайный лес**. Строим ансамбль алгоритмов по следующей схеме:

- Для построения i -го дерева:
 - Сначала, как в обычном бэггинге, из обучающей выборки X выбирается с возвращением случайная подвыборка X^i того же размера, что и X .
 - В процессе обучения каждого дерева в **каждой вершине** случайно выбираются $n < N$ признаков, где N — полное число признаков (метод случайных подпространств), и среди них ищется оптимальное разбиение. Такой приём позволяет управлять степенью коррелированности базовых алгоритмов.
- Чтобы получить предсказание ансамбля на тестовом объекте, усредняем отдельные ответы деревьев (для регрессии) или берём самый популярный класс (для классификации).

Random Forest (случайный лес) — комбинация бэггинга и метода случайных подпространств над решающими деревьями.

Основные параметры при построении случайного леса

- Глубина деревьев в случайном лесу

Ошибки модели (на которую мы можем повлиять) состоят из смещения и разброса. Разброс уменьшаем с помощью процедуры бэггинга. У неглубоких деревьев малое число параметров, то есть дерево способно запомнить только верхнеуровневые статистики обучающей подвыборки (низкая дисперсия, высокое смещение). Глубокие деревья запоминают подвыборку подробно (высокая дисперсия, низкое смещение).

Вывод: используем глубокие деревья.

- Число признаков при разбиении вершины в процессе обучения

Управляет качеством случайного леса. Чем больше признаков, тем больше корреляция между деревьями и тем меньше эффект от ансамблирования. Чем меньше признаков, тем слабее деревья. Практическая рекомендация — брать корень из числа всех признаков для классификации и треть признаков для регрессии.

- Число деревьев в случайном лесу

Можно построить график ошибки от числа деревьев и ограничить размер леса в тот момент, когда ошибка перестанет значимо уменьшаться.

35. Ансамбли классификаторов. Бустинг (boosting). Градиентный бустинг над решающими деревьями.

Ансамбли — это контролируемые алгоритмы обучения, которые комбинируют прогнозы из двух и более алгоритмов машинного обучения для построения более точных результатов. Результаты можно комбинировать с помощью голосования или усреднения. Первое зачастую применяется в классификации, а второе — в регрессии.

Существует 3 основных типа ансамблевых алгоритмов:

- Бэггинг.** Алгоритмы обучаются и работают параллельно на разных тренировочных наборах одного размера. Затем все они тестируются на одном наборе данных, а конечный результат определяется с помощью голосования.
- Бустинг.** В этом типе алгоритмы обучаются последовательно, а конечный результат отбирается с помощью голосования с весами.
- Стекинг (наложение).** Исходя из названия, этот подход состоит из двух уровней, расположенных друг на друге. Базовый представляет собой комбинацию алгоритмов, а верхний — мета-алгоритмы, основанные на базовом уровне.

Бустинг (boosting)

Бустинг (boosting) — это ансамблевый метод, в котором строится множество базовых алгоритмов из одного семейства слабых предсказывающих моделей, объединяющихся затем в более сильную модель. Отличие состоит в том, что в бэггинге и случайному лесу базовые алгоритмы учатся независимо и параллельно, а в бустинге — последовательно. Каждый следующий базовый алгоритм в бустинге обучается так, чтобы уменьшить общую ошибку всех своих предшественников, т.е. ошибку текущего ансамбля.

AdaBoost (Adaptive Boosting)

Алгоритм, предложенный Йоавом Фройндом и Робертом Шапире, может использоваться в сочетании с несколькими алгоритмами классификации для улучшения их эффективности за счет объединения их в ансамбль. Является адаптивным в том смысле, что каждый следующий ансамбль классификаторов строится по объектам, неверно классифицированным предыдущими комитетами.

AdaBoost — алгоритм построения «сильного» классификатора

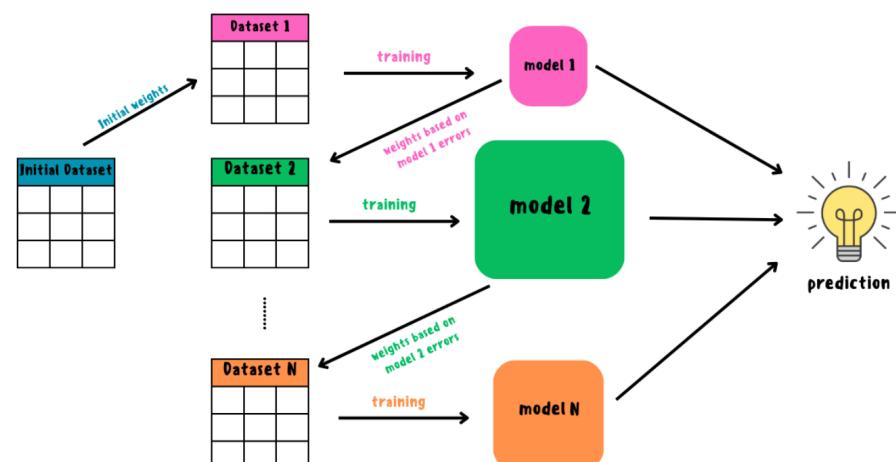
$$H(x) = \text{sign}(f(x)),$$

где

$$f(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

является линейной комбинацией «слабых» классификаторов

$$h_t(x): \chi \rightarrow \{-1, +1\}.$$



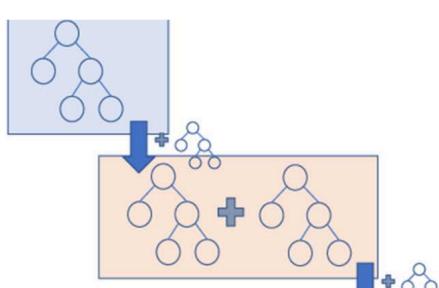
Основная идея

AdaBoost вызывает слабые классификаторы в цикле $t = 1, \dots, T$. После каждого вызова обновляется распределение весов D_t , которые отвечают за важность объектов обучающего множества для классификации. На каждой итерации веса каждого неверно классифицированного объекта возрастают, таким образом, новый комитет классификаторов «фокусирует своё внимание» на этих объектах.

Градиентный бустинг над решающими деревьями

Бустинг, использующий деревья решений в качестве базовых алгоритмов — **Gradient Boosting on Decision Trees (GBDT)**. Отлично работает на выборках с «табличными», неоднородными данными. Такой бустинг способен эффективно находить нелинейные зависимости в данных различной природы. Широко применяется во многих конкурсах по машинному обучению и задачах из индустрии (поисковом ранжировании, рекомендательных системах, таргетировании рекламы, предсказании погоды, пункта назначения такси и многих других).

Ошибка



Основная цель бустинга — уменьшение смещения. В качестве базовых алгоритмов часто выбирают алгоритмы с высоким смещением и небольшим разбросом. Например, глубина деревьев должна быть небольшой — обычно не больше 2-3 уровней. Важной причиной для выбора таких моделей — скорость обучения.

36. Обучение с подкреплением (reinforcement learning). Постановка задачи. Exploration vs Exploitation.

Обучение с подкреплением

Обучение с подкреплением (*reinforcement learning*) - способ машинного обучения, при котором испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Роль объектов играют пары «ситуация, принятное решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды). При обучении с подкреплением, в отличии от обучения с учителем, не предоставляются верные пары "входные данные - ответ", а принятие субоптимальных решений (дающих локальный экстремум) не ограничивается явно.

Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний (*exploration vs exploitation*).

Примеры прикладных задач: формирование инвестиционных стратегий, автоматическое управление технологическими процессами, самообучение роботов, и т.д.

Осуществляется поиск субоптимальных решений или стратегий того, как агент должен действовать в окружении, чтобы максимизировать некоторый долговременный выигрыш.

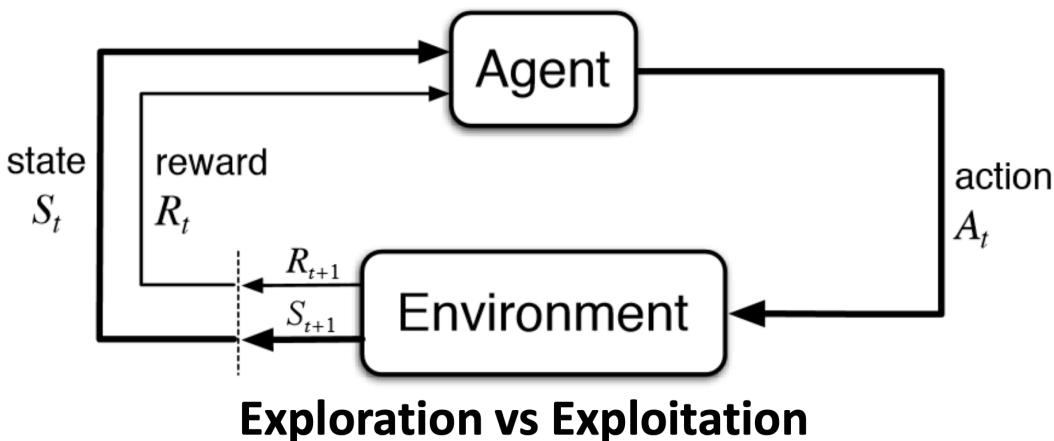
Маленький мальчик заходит в парикмахерскую. Парикмахер сразу же его узнаёт и говорит своим клиентам: «Смотрите, это самый глупый мальчик среди всех на свете! Сейчас я вам докажу!». В одной руке парикмахер держит доллар, в другой 25 центов. Зовёт мальчика, тот подходит и выбирает 25 центов. Все смеются, мальчик уходит. По дороге обратно, мальчика догоняет один из смеявшихся и спрашивает:
— А почему всё-таки ты выбрал 25 центов, а не 1 доллар?
— Потому что в тот день, когда я выберу 1 доллар, игра будет окончена.

Необходимые термины в Reinforcement Learning

- **Агент (agent):** Наша система, которая выполняет действия в среде, чтобы получить некоторую награду.
- **Среда (environment, e):** сценарий/окружение, с которым должен взаимодействовать агент.
- **Награда (reward, R):** немедленный возврат, который предоставляется агенту, после выполнения определенного действия или задачи. Является положительной или отрицательной величиной.
- **Состояние (state, s):** Состояние относится к текущему положению, возвращаемому средой.
- **Политика (policy, π):** стратегия, которая применяется агентом для принятия решения о следующем действии на основе текущего состояния.
- **Стоймость (value, V):** награда, которая ожидается в долгосрочной перспективе. По сравнению с краткосрочным вознаграждением, принимаем во внимание скидку (discount).
- **Функция полезности состояния (value function):** определяет размер переменной, которой является общая сумма награды.
- **Модель среды (Model of the environment):** имитатор поведения окружающей среды (демо-версия модели). Помогает определить, как будет вести себя среда.
- **Значение Q или значение полезности действия (Q):** значение Q очень похоже на value (V). Главное различие в том, что Q принимает текущее действие в качестве дополнительного параметра.

Простейшая постановка задачи

- На каждом шаге агент может находиться в состоянии $s \in S$.
- На каждом шаге агент выбирает из имеющегося набора действий некоторое действие $a \in A$.
- Окружающая среда сообщает агенту, какую награду r он за это получил и в каком состоянии s' после этого оказался.



- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Та или иная стратегия может быть хороша, но вдруг она не оптимальная? Как достичь оптимального соотношения между исследованием нового и использованием имеющихся знаний?
- Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний, т.е. **exploration vs exploitation**. Эта проблематика всегда присутствует в обучении с подкреплением.

37. Марковские процессы. Постановка задачи обучения с подкреплением. Подход к решению.

Обучение с подкреплением

Обучение с подкреплением (*reinforcement learning*) - способ машинного обучения, при котором испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Роль объектов играют пары «ситуация, принятное решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды). При обучении с подкреплением, в отличии от обучения с учителем, не предоставляются верные пары "входные данные - ответ", а принятие субоптимальных решений (дающих локальный экстремум) не ограничивается явно.

Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний (**exploration vs exploitation**).

Маленький мальчик заходит в парикмахерскую. Парикмахер сразу же его узнаёт и говорит своим клиентам: «Смотрите, это самый глупый мальчик среди всех на свете! Сейчас я вам докажу!». В одной руке парикмахер

Марковские процессы

Марковский процесс — случайный процесс, эволюция которого после любого заданного значения временного параметра t не зависит от эволюции, предшествовавшей t , при условии, что значение процесса в этот момент фиксировано («будущее» процесса зависит от «прошлого» лишь через «настоящее»).

Марковское свойство — в теории вероятностей и статистике термин, который относится к памяти случайного процесса.

Стochastic process обладает марковским свойством, если условное распределение вероятностей будущих состояний процесса зависит только от нынешнего состояния, а не от последовательности событий, которые предшествовали этому. Процесс, обладающий этим свойством, называется марковским процессом.

Для марковских цепей с дискретным временем. В случае, если S является дискретным множеством состояний и время дискретно, то марковское свойство может быть сформулировано следующим образом:

$$\begin{aligned}\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_0 = x_0) = \\ \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})\end{aligned}$$

Марковский процесс принятия решений (МППР) *Markov decision process (MDP)*

Спецификация задачи последовательного принятия решений для полностью наблюдаемой среды с марковской моделью перехода и дополнительными вознаграждениями. Слово марковский в названии отражает выполнение марковского свойства для таких процессов. Такой процесс служит математической основой для моделирования последовательного принятия решений в ситуациях, где результаты частично случайны и частично под контролем лица, принимающего решения.

Эта спецификация используется во множестве областей, включая робототехнику, автоматизированное управление, экономику, производство, а также в качестве основы обучения с подкреплениями.

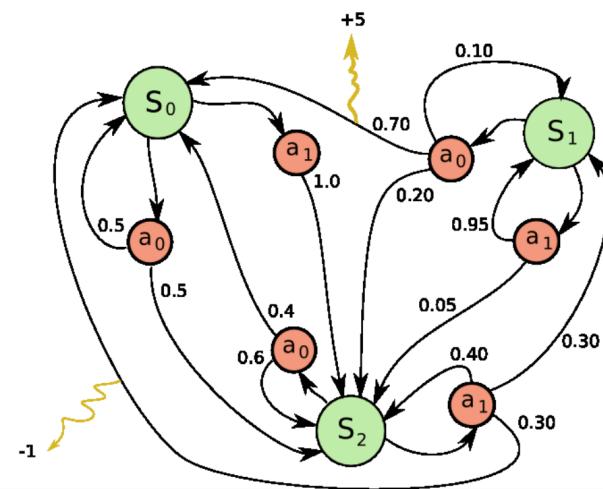
МППР - Определение

Чтобы определить марковский процесс принятия решений, нужно задать 4-кортеж $(S, A, P(\cdot, \cdot), R(\cdot, \cdot))$, где:

- S - конечное множество состояний;
- A - конечное множество действий (часто представляется в виде множеств A_s , действий доступных из состояния s);
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ - вероятность, что действие a в состоянии s во время t приведет в состояние s' ко времени $t + 1$;

- $R_a(s, s')$ - вознаграждение, получаемое после перехода в состояние s' из состояния s , при совершении действия a .

Стратегия π — функция (в общем случае распределение вероятностей), сопоставляющая состоянию действие. Такой Марковский процесс принятия решений можно рассматривать, как Марковскую цепь.



МППР - Цель оптимизации

Решить марковский процесс принятия решений означает найти стратегию, максимизирующую "вознаграждение" (функцию ценности) - оптимальную стратегию. Самая простая функция ценности это математическое ожидание формального ряда:

$$E \left[\sum_{t=0}^{\infty} R_{a_t}(s_t, s_{t+1}) \right]$$

где $a_t = \pi(s_t)$, а математическое ожидание берётся в соответствии с $s_{t+1} \sim P_{a_t}(s_t, \cdot)$. Такую функцию можно использовать если гарантируется, что ряд сходится, а значит наличие терминального состояния, где $P_a(s, s) = 1$ и $R_a(s, s) = 0$.

Если же сходимость ряда не гарантируется, то обычно делают одно из двух:

- Рассматривают только конечное число слагаемых:

$$E \left[\sum_{t=0}^N R_{a_t}(s_t, s_{t+1}) \right]$$

- Вводят коэффициент обесценивания (дисконтирования) $\gamma \in [0, 1]$:

$$E \left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \right]$$

На практике второй вариант более гибкий, так как учитывает более долгосрочную перспективу и чаще используется именно он.

МППР - Функции полезности

Для максимизации ряда $E \left[\sum_{t=0}^{\infty} R_{a_t}(s_t, s_{t+1}) \right]$ вводят две функции полезности:

- Функция полезности состояния:

$$V_{\pi}(s) = E \left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | s_0 = s, a_t = \pi(s_t) \right]$$

- Функция полезности действия:

$$Q_{\pi}(s, a) = E \left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | s_0 = s, a_0 = a, a_t = \pi(s_t) \forall t \geq 1 \right]$$

где математическое ожидание берётся в соответствии с $s_{t+1} \sim P_{a_t}(s_t, \cdot)$.

А также их максимумы по всем стратегиям:

$$V_*(s) = \max_{\pi} V_{\pi}(s) \text{ и } Q_*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$

Можно доказать, что эти функции также являются функциями полезности состояния и полезности действия соответственно, а также, что они достигаются на детерминированной стратегии. Заметим, что по функции Q_* можно восстановить её стратегию, которая будет оптимальной.

Для определения оптимальной стратегии используется отношение порядка на множестве стратегий.

$$\pi_1 \leq \pi_2 \Leftrightarrow \forall V_{\pi_1}(s) \leq V_{\pi_2}(s), s \in S$$

Наибольшая стратегия называется оптимальной.

Постановка задачи обучения с подкреплением

Составные части:

- множество состояний среды (*states*) S ;
- множество действий (*actions*) A ;
- множество вещественнозначных скалярных "выигрышей" (*rewards*) R .

Игра агента со средой:

- инициализация стратегии $\pi_1(a|s)$ и состояния среды s_1
- для всех $t = 1 \dots T$:
 - агент выбирает действие $a_t \sim \pi_1(a|s_t)$
 - среда генерирует награду $r_{t+1} \sim p(r|a_t, s_t)$ и новое состояние $s_{t+1} \sim p(s|a_t, s_t)$
 - агент корректирует стратегию $\pi_{t+1}(a|s)$

Таким образом в произвольный момент времени t агент характеризуется состоянием $s_t \in S$ и множеством возможных действий $A(s_t)$. Выбирая действие $a \in A(s_t)$, он переходит в состояние s_{t+1} и получает выигрыш r_{t+1} . Основываясь на таком взаимодействии с окружающей средой, агент, обучающийся с подкреплением, должен выработать стратегию $\pi: S \rightarrow A$, которая максимизирует величину:

- $R = r_0 + r_1 + \dots + r_n$ в случае марковского процесса принятия решений (МППР), имеющего терминальное состояние;
- $R = \sum_t \gamma^t r_t$ для МППР без терминальных состояний (где $0 \leq \gamma \leq 1$ — дисконтирующий множитель для "предстоящего выигрыша").

Марковское свойство (МППР):

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_1, a_1) = P(s_{t+1} = s', r_{t+1} = r | s_t, a_t)$$

МППР называется финитным, если $|A| < \infty$, $|S| < \infty$.

Таким образом, обучение с подкреплением особенно хорошо подходит для решения задач, связанных с выбором между долгосрочной и краткосрочной выгодой.

Подход к решению

Наивный подход к решению этой задачи подразумевает следующие шаги:

- опробовать все возможные стратегии;
- выбрать стратегию с наибольшим ожидаемым выигрышем.

Проблемы:

- количество доступных стратегий может быть велико или бесконечно;
- выигрыши стохастические — чтобы точно оценить выигрыш от каждой стратегии потребуется многократно применить каждую из них.

Решения:

- оценка функций полезности;
- прямая оптимизация стратегий.

Подход с использованием функции полезности использует множество оценок ожидаемого выигрыша только для одной стратегии π (либо текущей, либо оптимальной). При этом пытаются оценить либо ожидаемый выигрыш, начиная с состояния s , при дальнейшем следовании стратегии π ,

$$V(s) = E[R|s, \pi],$$

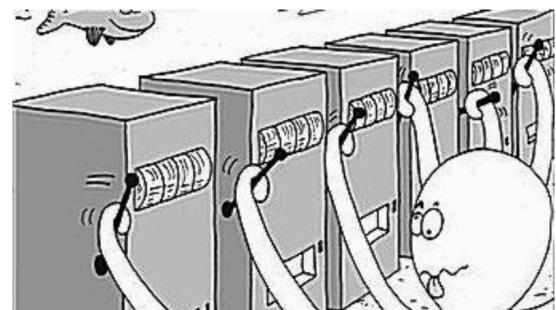
либо ожидаемый выигрыш, при принятии решения a в состоянии s и дальнейшем соблюдении π ,

$$Q(s, a) = E[R|s, \pi, a].$$

38. Задача о многоруком бандите (The multi-armed bandit problem). Стратегии.

Задача о многоруком бандите (*The multi-armed bandit problem*)

- Агенты с одним состоянием, т.е. состояние агента не меняется. У него фиксированный набор действий и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша.
- Нужно заработать побольше:
Exploration vs. Exploitation
(разведка против эксплуатации).
- Жадные и ϵ -жадные стратегии
(greedy & ϵ -greedy)



Exploration vs Exploitation

- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Та или иная стратегия может быть хороша, но вдруг она не оптимальная? Как достичь оптимального соотношения между исследованием нового и использованием имеющихся знаний?
- Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний, т.е. *exploration vs exploitation*. Эта проблематика всегда присутствует в обучении с подкреплением.

Задача о многоруком бандите - Формулировка

A — множество возможных действий (ручек автомата),

$p_a(r)$ — неизвестное распределение награды $r \in R \quad \forall a \in A$,

$\pi_t(a)$ — стратегия агента в момент $t \quad \forall a \in A$.

Игра агента со средой:

- инициализация стратегии $\pi_1(a)$;
- для всех $t = 1 \dots T$:
 - агент выбирает действие (ручку) $a_t \sim \pi_t(a)$;
 - среда генерирует награду $r_t \sim p_{a_t}(r)$;
 - агент корректирует стратегию $\pi_{t+1}(a)$.

Средняя награда в t играх: $Q_t(a) = \frac{\sum_{i=1}^t r_i[a_i=a]}{\sum_{i=1}^t [a_i=a]} \rightarrow \max$,

Ценность действия a : $Q^*(a) = \lim_{t \rightarrow \infty} Q_t(a) \rightarrow \max$

N -рукий бандит - на каждом шаге выбираем за какую из N ручек автомата дернуть. Каждому действию соответствует некоторое распределение (не меняется со временем). Если распределения известны, то стратегия заключается в том, чтобы подсчитать математическое ожидание для каждого из распределений, выбрать действие с максимальным математическим ожиданием и совершать это действие на каждом шаге. Проблема, что распределения неизвестны, однако можно оценить математическое ожидание случайной величины ξ с неизвестным распределением.

$$E(\xi) = \frac{1}{K} \sum_{k=1}^K \xi_k$$

Жадная (greedy) стратегия

Начальные значения:

- $P_a = 0$ для $\forall a \in \{1 \dots N\}$ — сколько раз было выбрано действие a ,
- $Q_a = 0$ для $\forall a \in \{1 \dots N\}$ — текущая оценка математического ожидания награды для действия a

На каждом шаге t :

- Выбираем действие с максимальной оценкой математического ожидания:
$$a_t = \operatorname{argmax}_{a \in A} Q_a$$
- Выполняем действие a_t и получаем награду $R(a_t)$;
- Обновляем оценку математического ожидания для действия a_t :

$$\begin{aligned} P_{a_t} &= P_{a_t} + 1 \\ Q_{a_t} &= Q_{a_t} + \frac{1}{P_{a_t}} (R(a_t) - Q_{a_t}) \end{aligned}$$

В чем проблема?

Пусть у нас есть "двурукий" бандит. Первая ручка всегда выдаёт награду равную 1, вторая всегда выдаёт 2. Действуя согласно жадной стратегии мы дёрнем в начале первую ручку, так как в начале оценки математических ожиданий равны нулю, увеличим её оценку до $Q_1 = 1$. В дальнейшем всегда будем выбирать первую ручку, а значит на каждом шаге будем получать на 1 меньше, чем могли бы.

ϵ -жадная (ϵ -greedy) стратегия

Введем параметр $\epsilon \in (0,1)$.

На каждом шаге t :

- Получим значение a - случайной величины равномерно распределенной на отрезке $(0,1)$;
- Если $a \in (0, \epsilon)$, то выберем действие $a_t \in A$ случайно и равновероятно, иначе как в жадной стратегии выберем действие с максимальной оценкой математического ожидания;
- Обновляем оценки так же как в жадной стратегии.

Если $\epsilon = 0$, то это обычная жадная стратегия. Однако если $\epsilon > 0$, то в отличии от жадной стратегии на каждом шаге с вероятностью ϵ происходит "исследование" случайных действий.

39. Марковские процессы. Агент, среда, обратная связь, состояние, функции ценности состояния (Value function), качества действия (Q-function);

Марковские процессы

Марковский процесс — случайный процесс, эволюция которого после любого заданного значения временного параметра t не зависит от эволюции, предшествовавшей t , при условии, что значение процесса в этот момент фиксировано («будущее» процесса зависит от «прошлого» лишь через «настоящее»).

Марковское свойство — в теории вероятностей и статистике термин, который относится к памяти случайного процесса.

Стохастический процесс обладает марковским свойством, если условное распределение вероятностей будущих состояний процесса зависит только от нынешнего состояния, а не от последовательности событий, которые предшествовали этому. Процесс, обладающий этим свойством, называется марковским процессом.

Для марковских цепей с дискретным временем. В случае, если S является дискретным множеством состояний и время дискретно, то марковское свойство может быть сформулировано следующим образом:

$$\begin{aligned}\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_0 = x_0) = \\ \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})\end{aligned}$$

Необходимые термины в Reinforcement Learning

- **Агент (agent):** Наша система, которая выполняет действия в среде, чтобы получить некоторую награду.
- **Среда (environment, e):** сценарий/окружение, с которым должен взаимодействовать агент.
- **Награда (reward, R):** немедленный возврат, который предоставляется агенту, после выполнения определенного действия или задачи. Является положительной или отрицательной величиной.
- **Состояние (state, s):** Состояние относится к текущему положению, возвращаемому средой.
- **Политика (policy, π):** стратегия, которая применяется агентом для принятия решения о следующем действии на основе текущего состояния.
- **Стоимость (value, V):** награда, которая ожидается в долгосрочной перспективе. По сравнению с краткосрочным вознаграждением, принимаем во внимание скидку (discount).
- **Функция полезности состояния (value function):** определяет размер переменной, которой является общая сумма награды.
- **Модель среды (Model of the environment):** имитатор поведения окружающей среды (демо-версия модели). Помогает определить, как будет вести себя среда.
- **Значение Q или значение полезности действия (Q):** значение Q очень похоже на value (V). Главное различие в том, что Q принимает текущее действие в качестве дополнительного параметра.

40. Эволюционный алгоритм (Evalutionary algorithm). Основные этапы. Преимущества и недостатки.

Эволюционный алгоритм Evalutionary algorithm



В искусственном интеллекте и машинном обучении эволюционные алгоритмы — это раздел эволюционных вычислений (моделирования), в которых используются модели процессов естественного отбора (размножение, рекомбинация, мутация и отбор) и принципы природной эволюции для решения задач оптимизации.

Решения задачи рассматриваются как особи популяции (агенты или индивидуумы). Множество особей принято называть популяцией. Качество каждого решения оценивается с помощью специальной целевой функции приспособленности (fitness function — фитнес-функция), которая задается окружающей средой. Такие алгоритмы относятся к адаптивным поисковым механизмам, включают эволюцию популяции и содержат следующие шаги:

1. Формируется начальная популяция методом случайного отбора (первое поколение).
2. Оценивается пригодность каждого члена популяции с помощью фитнес-функции.
3. Повторяются следующие действия (эволюция):
 - **отбор** — выбор наиболее приспособленных особей для размножения (родители);
 - **размножение** — формирование новых особей путем скрещивания и мутации, а затем оценка их пригодности;
 - **рекомбинация** — наименее приспособленные особи предыдущего поколения заменяются наиболее приспособленными особями нового поколения.

Преимущества:	Недостатки:
<ul style="list-style-type: none">• применимы к широкому классу задач оптимизации;• легко комбинируются с другими методами;• позволяют получать хорошо интерпретируемые результаты;• обладают интерактивностью — можно включить в популяцию предложенные пользователем решения;• рассматривают большое количество альтернативных решений.	<ul style="list-style-type: none">• отсутствие гарантии нахождения оптимального решения за конечное время — алгоритм реализует локальную оптимизацию;• слабая теоретическая база — алгоритм является эвристическим, т.е. точность и строгость постановки приносятся в жертву реализуемости;• высокие вычислительные затраты.

41. Генетический алгоритм. Основные этапы. Целевая функция приспособленности (fitness function)

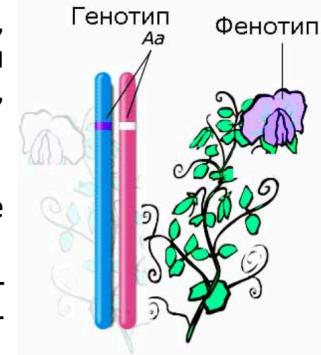
Генетический алгоритм (ГА)

Эвристический алгоритм поиска, предложенный в 1975 году Джоном Холландом (John Holland) из Мичиганского университета, используется для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений

Работает с кодовыми последовательностями (КП), безотносительно к их смысловой интерпретации, поэтому КП и ее структура задается понятием **генотипа**, а его интерпретация, с точки зрения решаемой задачи, понятием **фенотипа**.

- **Фенотип** определяет, чем является объект в реальном мире.
- **Генотип** содержит всю информацию об объекте на уровне хромосомного набора.

Каждый ген, то есть элемент информации генотипа, имеет свое отражение в фенотипе, а совокупность генов называют хромосомой.



Каждая КП представляет, по сути, точку пространства поиска и называется особью или индивидуумом. Набор КП (особей) образует исходное множество решений K (популяцию). Количество особей в популяции характеризуется ее размером.

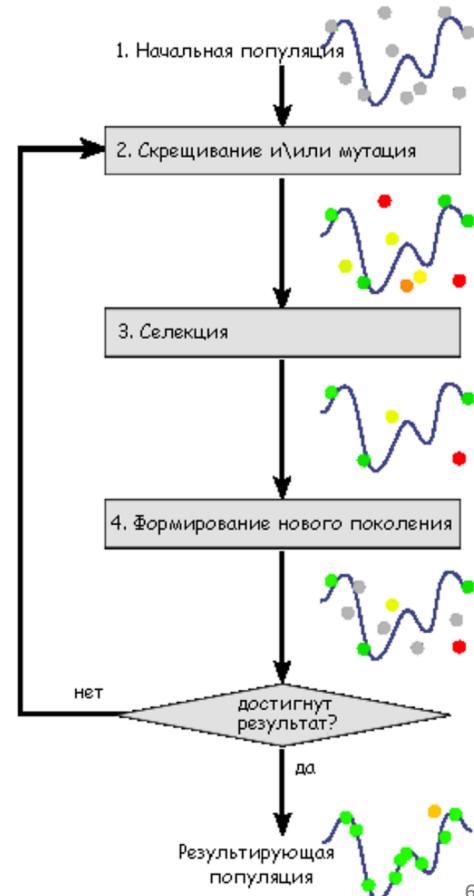
Особи в популяции оцениваются с использованием «функции приспособленности» (fitness function), в результате чего с каждым генотипом ассоциируется значение, которое определяет, насколько хорошо им описываемый фенотип решает поставленную задачу.

Основные этапы ГА

1. Задать целевую функцию приспособленности (fitness function) для особей популяции
2. Создать начальную популяцию
3. Начало цикла эволюции
 - Выбор родительских особей
 - «Скрещивание» (Crossover)
 - «Мутация» (Mutation) / Инверсия
 - Вычисление значений fitness function
 - Формирование нового поколения (селекция)
 - Если выполняются условия остановки, то «конец эволюционного процесса», иначе переход в начало цикла эволюции.

Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий остановки алгоритма. Критерием может быть:

- нахождение глобального, либо субоптимального решения;
- исчерпание числа поколений, отпущеных на эволюцию;
- исчерпание времени, отпущеного на эволюцию.



Целевая функция приспособленности (fitness function)

Вещественная или целочисленная функция одной или нескольких переменных, подлежащая оптимизации в результате работы генетического алгоритма, направляет эволюцию в сторону оптимального решения. Позволяет оценить степень приспособленности конкретных особей в популяции.

На выбор (построение) фитнесс-функции оказывают влияние следующие факторы:

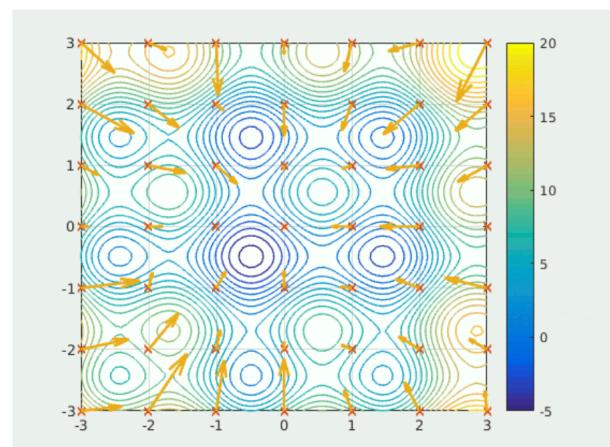
- тип задачи – максимизация или минимизация;
- содержание шумов окружающей среды в фитнесс-функции;
- возможность динамического изменения фитнесс-функции в процессе решения задачи;
- объем допустимых вычислительных ресурсов;
- насколько различные значения для близких особей должна давать фитнесс-функция для упрощения отбора родительских особей;
- должна ли она содержать ограничения решаемой задачи;
- может ли она совмещать различные подцели (например, для многокритериальных задач).

В ГА фитнесс-функция может использоваться в виде черного ящика, т.е. для данной хромосомы она вычисляет значение, определяющее качество данной особи, и при этом внутри может быть реализована: в виде математической функции, программы моделирования (в том числе имитационного), нейронной сети, экспертной оценки и др.

42. Алгоритм роя частиц. Инициализация, выполнение.

Классический алгоритм роя частиц

Метод роя частиц (МРЧ) был предложен Кеннеди, Эберхартом и Ши (1995 год) и изначально предназначался для имитации социального поведения. Алгоритм моделирует многоагентную систему, где агенты-частицы (популяция возможных решений) двигаются к оптимальным решениям, обмениваясь при этом информацией с соседями.



Текущее состояние частицы характеризуется координатами в пространстве решений (то есть, собственно, связанным с ними решением), а также вектором скорости перемещения. Оба этих параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит координаты лучшего из найденных ей решений, а также лучшее из пройденных всеми частицами решений – этим имитируется мгновенный обмен информацией между птицами.

Алгоритм роя частиц - инициализация

Пусть $f: \mathbb{R}^n \rightarrow \mathbb{R}$ — целевая функция, которую требуется минимизировать, S — количество частиц в рое, каждой из которых сопоставлена координата $x_i \in \mathbb{R}^n$ в пространстве решений и скорость $v_i \in \mathbb{R}^n$. Пусть также p_i — лучшее из известных положений частицы с индексом i , а g — наилучшее известное состояние роя в целом.



Для каждой частицы $i = 1, \dots, S$:

- Сгенерировать начальное положение частицы с помощью случайного вектора $x_i \sim U(b_{lo}, b_{up})$, имеющего многомерное равномерное распределение, где b_{lo}, b_{up} — нижняя и верхняя границы пространства решений соответственно.
- Присвоить лучшему известному положению частицы его начальное значение: $p_i = x_i$.
- Если $f(p_i) < f(g)$, то обновить наилучшее известное состояние роя: $g = p_i$.
- Присвоить значение скорости частицы:

$$v_i \sim U(-(b_{up} - b_{lo}), (b_{up} - b_{lo})).$$

Алгоритм роя частиц - выполнение

- Пока не выполнен критерий остановки (например, достижение заданного числа итераций или необходимого значения целевой функции), повторять для каждой частицы $i = 1, \dots, S$ следующие действия:
 - Сгенерировать случайные векторы $r_p, r_g \sim U(0,1)$.
 - На каждой итерации направление и длина вектора скорости каждой из частиц изменяются в соответствие со сведениями о найденных оптимумах:
 - Обновить скорость частицы: $v_{i,t+1} = \omega v_{i,t} + \varphi_p r_p \odot (p_i - x_{i,t}) + \varphi_g r_g \odot (g - x_{i,t})$, где операция \odot — покомпонентное умножение, φ_p, φ_g — постоянные ускорения, ω — коэффициент инерции.
 - Обновить положение частицы x_i на вектор скорости: $x_{i,t+1} = x_{i,t} + v_{i,t+1}$. Этот шаг выполняется вне зависимости от улучшения значения целевой функции.
 - Если $f(x_i) < f(p_i)$, то:
 - Обновить лучшее известное положение частицы: $p_i = x_i$.
 - Если $f(p_i) < f(g)$, то лучшее известное состояние роя: $g = p_i$

➤ g содержит лучшее из найденных решений.

Параметры $\omega, \varphi_p, \varphi_g$ выбираются вычислителем и определяют поведение и эффективность метода в целом. Эти параметры составляют предмет многих исследований.

43. Эволюционные алгоритмы. Алгоритм пчелиной колонии.

Эволюционный алгоритм Evolutionary algorithm



В искусственном интеллекте и машинном обучении эволюционные алгоритмы — это раздел эволюционных вычислений (моделирования), в которых используются модели процессов естественного отбора (размножение, рекомбинация, мутация и отбор) и принципы природной эволюции для решения задач оптимизации.

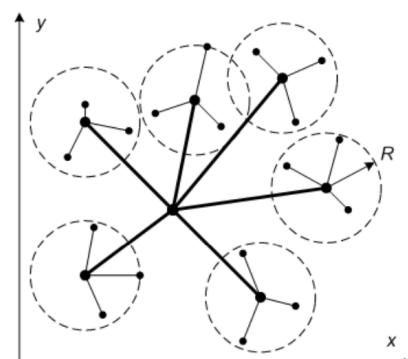
Решения задачи рассматриваются как особи популяции (агенты или индивидуумы). Множество особей принято называть популяцией. Качество каждого решения оценивается с помощью специальной целевой функции приспособленности (fitness function — фитнес-функция), которая задается окружающей средой. Такие алгоритмы относятся к адаптивным поисковым механизмам, включают эволюцию популяции и содержат следующие шаги:

1. Формируется начальная популяция методом случайного отбора (первое поколение).
2. Оценивается пригодность каждого члена популяции с помощью фитнес-функции.
3. Повторяются следующие действия (эволюция):
 - **отбор** — выбор наиболее приспособленных особей для размножения (родители);
 - **размножение** — формирование новых особей путем скрещивания и мутации, а затем оценка их пригодности;
 - **рекомбинация** — наименее приспособленные особи предыдущего поколения заменяются наиболее приспособленными особями нового поколения.

Преимущества:	Недостатки:
<ul style="list-style-type: none">• применимы к широкому классу задач оптимизации;• легко комбинируются с другими методами;• позволяют получать хорошо интерпретируемые результаты;• обладают интерактивностью — можно включить в популяцию предложенные пользователем решения;• рассматривают большое количество альтернативных решений.	<ul style="list-style-type: none">• отсутствие гарантии нахождения оптимального решения за конечное время — алгоритм реализует локальную оптимизацию;• слабая теоретическая база — алгоритм является эвристическим, т.е. точность и строгость постановки приносятся в жертву реализуемости;• высокие вычислительные затраты.

Алгоритм пчелиной колонии

Алгоритм оптимизации подражанием пчелиной колонии (*artificial bee colony optimization, ABC*) — один из полиномиальных эвристических алгоритмов для решения дискретных (комбинаторных) и непрерывных задач глобальной оптимизации в области информатики и исследования операций. Относится к категории стохастических бионических алгоритмов, основан на имитации поведения колонии медоносных пчел при сборе нектара в природе. Предложен Д. Карабога в 2005 г.



Алгоритм включает два основных этапа:

- При инициализации (начальной разведке) производится выполнение разведки пространства признаков с целью определения K его наиболее перспективных точек с наилучшими значениями целевой функции $f(X) = f(x_1, x_2, \dots, x_n)$ (в простейшем случае с использованием метода случайного перебора), которые запоминаются в улье.
- Последующая работа пчел улья. В окрестностях выбранных точек производится локальная разведка в пределах заданного радиуса разведки R с целью попытки уточнения решения, при этом при достижении улучшения в улье сохраняется обновленное значение f^* и соответствующий ему вектор параметров целевой функции X^* . Комбинируя работу пчел-разведчиков и рабочих пчел на протяжении заданного числа итераций C алгоритм обеспечивает постепенное улучшение запоминаемой выборки $\mathcal{R} = [X_1, X_2, \dots, X_K]$ из K решений.

По завершении его работы из указанного множества решений выбирается наилучшее, что является результатом работы алгоритма.

44. Эволюционные алгоритмы. Муравьиный алгоритм.

Эволюционный алгоритм Evolutionary algorithm



В искусственном интеллекте и машинном обучении эволюционные алгоритмы — это раздел эволюционных вычислений (моделирования), в которых используются модели процессов естественного отбора (размножение, рекомбинация, мутация и отбор) и принципы природной эволюции для решения задач оптимизации.

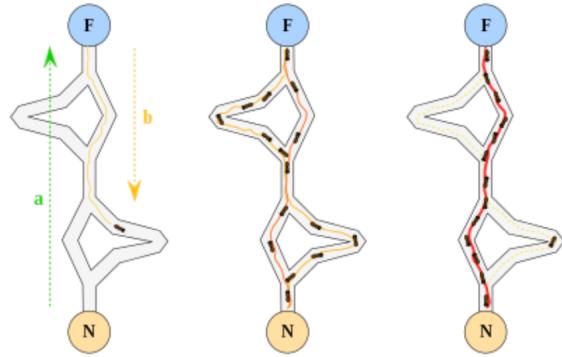
Решения задачи рассматриваются как особи популяции (агенты или индивидуумы). Множество особей принято называть популяцией. Качество каждого решения оценивается с помощью специальной целевой функции приспособленности (fitness function — фитнес-функция), которая задается окружающей средой. Такие алгоритмы относятся к адаптивным поисковым механизмам, включают эволюцию популяции и содержат следующие шаги:

1. Формируется начальная популяция методом случайного отбора (первое поколение).
2. Оценивается пригодность каждого члена популяции с помощью фитнес-функции.
3. Повторяются следующие действия (эволюция):
 - **отбор** — выбор наиболее приспособленных особей для размножения (родители);
 - **размножение** — формирование новых особей путем скрещивания и мутации, а затем оценка их пригодности;
 - **рекомбинация** — наименее приспособленные особи предыдущего поколения заменяются наиболее приспособленными особями нового поколения.

Преимущества:	Недостатки:
<ul style="list-style-type: none">• применимы к широкому классу задач оптимизации;• легко комбинируются с другими методами;• позволяют получать хорошо интерпретируемые результаты;• обладают интерактивностью — можно включить в популяцию предложенные пользователем решения;• рассматривают большое количество альтернативных решений.	<ul style="list-style-type: none">• отсутствие гарантии нахождения оптимального решения за конечное время — алгоритм реализует локальную оптимизацию;• слабая теоретическая база — алгоритм является эвристическим, т.е. точность и строгость постановки приносятся в жертву реализуемости;• высокие вычислительные затраты.

Муравьиный алгоритм

Муравьиный алгоритм (алгоритм оптимизации подражанием муравьиной колонии - *ant colony optimization*, ACO) — алгоритм для нахождения приближенных решений задач оптимизации на графах, таких как задача коммивояжера, транспортная задача и аналогичных задач поиска маршрутов на графах. Вычислительные затраты алгоритма полиномиально зависят от объема исходных данных.



В основе метода лежит поведение муравьев некоторых видов, которые изначально перемещаются в поисках пищи случайным образом, но, найдя ее, возвращаются в свою колонию, помечая путь феромонами. Это избавляет других муравьев от необходимости случайного поиска пищи и делает его более целенаправленным: найдя путь, помеченный феромонами, муравьи движутся по нему, усиливая плотность феромонов.

Однако со временем плотность феромонов уменьшается. И происходит это тем быстрее, чем длиннее путь, поскольку интервал перемещения по нему муравьев велик. Напротив, чем путь короче и чем интенсивнее он используется, тем выше плотность феромона на нем. Таким образом, выбирая пути с наибольшей плотностью феромона, муравьи сокращают расстояние, проходимое в поисках пищи, что эквивалентно поиску кратчайшего пути на графике.

На коротком пути, для сравнения, прохождение будет более быстрым, и, как следствие, плотность феромонов остаётся высокой. Если бы феромоны не испарялись, то путь, выбранный первым, был бы самым привлекательным.

Муравьиный алгоритм

Работа начинается с размещения муравьёв в вершинах графа (городах), затем начинается движение муравьёв — направление определяется вероятностным методом, на основании формулы вида:

Рёбра: Муравей будет двигаться от узла i к узлу j с вероятностью:

$$p_{i,j} = \frac{(\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)}{\sum(\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)}$$

$\tau_{i,j}^\alpha$ - количество феромонов на ребре i, j ;

α - параметр, контролирующий влияние $\tau_{i,j}^\alpha$ (определяет «стадность» алгоритма);

$\eta_{i,j}^\beta$ - привлекательность ребра i, j (начальное значение $1/d_{i,j}$, где d расстояние);

β - параметр, контролирующий влияние $\eta_{i,j}^\beta$ (определяет «жадность» алгоритма);

$\alpha + \beta = 1$.

Обновление феромонов

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j}$$

$\tau_{i,j}$ - количество феромонов на ребре i, j

ρ - скорость испарения феромона,

$\Delta\tau_{i,j}$ - количество отложенного феромона, обычно определяется как:

$$\Delta\tau_{i,j}^k = \begin{cases} 1/L_k & \text{if Ant } k \text{ travels on edge } i, j \\ 0 & \text{otherwise} \end{cases}$$

где L_k -стоимость k -ого пути муравья (обычно длина).

45. Кластеризация. Цели, формальная постановка задачи, принципиальная неоднозначность.

Кластеризация (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Цели кластеризации

- **Улучшение понимания данных** за счет выявления структурных групп;
- **Классификация объектов.** Попытка понять зависимости между объектами путем выявления их кластерной структуры. Разбиение выборки на группы схожих объектов упрощает дальнейшую обработку данных, позволяет применить к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»). В данном случае стремятся уменьшить число кластеров для выявления наиболее общих закономерностей;
- **Сжатие данных.** Можно сократить размер исходной выборки, взяв один или несколько наиболее типичных представителей каждого кластера. Здесь важно наиболее точно очертить границы каждого кластера, их количество не является важным критерием;
- **Обнаружение новизны (обнаружение шума).** Выделение объектов, которые не подходят по критериям ни в один кластер. Обнаруженные объекты в дальнейшем обрабатывают отдельно.

Формальная постановка задачи

Пусть X — множество объектов, Y — множество идентификаторов (номеров, имён, меток) кластеров. Задана функция расстояний между объектами $\rho(x, x')$. Имеется конечная обучающая выборка объектов $X^m = \{x_1, \dots, x_m\} \subset X$. Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ , а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^m$ приписывается номер кластера y_i .

Алгоритм кластеризации — это функция $\alpha: X \rightarrow Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$. Множество Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Принципиальная неоднозначность

Решение задачи кластеризации принципиально неоднозначно:

- Не существует однозначно наилучшего критерия качества кластеризации.
- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием.
- Результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

46. Кластеризация. Основные этапы. Меры расстояний.

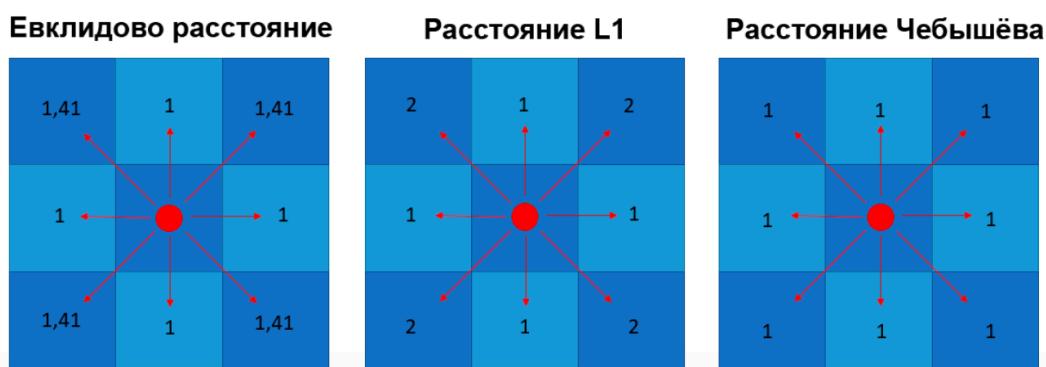
Кластеризация (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Основные этапы

- Отбор выборки объектов для кластерного анализа.
- Определение множества переменных, по которым будут оцениваться объекты в выборке, то есть признакового пространства.
- Выбор и вычисление значений мер сходства (или различия) между объектами.
- Применение метода кластерного анализа для создания групп сходных объектов.
- Проверка достоверности и представление результатов анализа.

Меры расстояний

- Евклидово расстояние: $\rho(x, x') = \sqrt{\sum_i^n (x_i - x'_i)^2}$
- Квадрат евклидова расстояния: $\rho(x, x') = \sum_i^n (x_i - x'_i)^2$
- Расстояние городских кварталов L1 (манхэттенское расстояние): $\rho(x, x') = \sum_i^n |x_i - x'_i|$
- Расстояние Чебышева: $\rho(x, x') = \max(|x_i - x'_i|)$
- Степенное расстояние: $\rho(x, x') = \sqrt[r]{\sum_i^n (x_i - x'_i)^r}$
где r и p – параметры, определяемые пользователем.



47. Кластеризация. Меры качества кластеризации.

Кластеризация (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Меры качества кластеризации

Задачу кластеризации можно ставить как задачу дискретной оптимизации, т.е. необходимо так присвоить номера кластеров y_i объектам x_i , чтобы значение выбранного функционала качества приняло наилучшее значение.

Функционал качества:

- Среднее внутрикластерное расстояние должно быть как можно меньше:
$$F_0 = \frac{\sum_{i < j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i = y_j]} \rightarrow \min$$
- Среднее межкластерное расстояние должно быть как можно больше:
$$F_1 = \frac{\sum_{i < j} [y_i \neq y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i \neq y_j]} \rightarrow \max$$

Если алгоритм кластеризации вычисляет центры кластеров μ_y , $y \in Y$, то можно определить функционалы:

- Сумма средних внутрикластерных расстояний $\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i: y_i = y} \rho^2(x_i, \mu_y) \rightarrow \min$ должна быть как можно меньше:
где $K_y = \{x_i \in X^l | y_i = y\}$ кластер с номером y , состоящий из $|K_y|$ точек.
- Сумма межкластерных расстояний должна быть как можно больше: $\Phi_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \rightarrow \max$
где μ - центр масс всей выборки.

На практике вычисляют отношение пары функционалов, чтобы сразу учесть как межкластерные, так и внутрикластерные расстояния:

$$\frac{F_0}{F_1} \rightarrow \min, \text{ или } \frac{\Phi_0}{\Phi_1} \rightarrow \min$$

Методы оценки качества кластеризации

Принято выделять две группы методов оценки качества кластеризации:

- Внешние** (англ. *External*) меры основаны на сравнении результата кластеризации с априори известным разделением на классы. Данные меры используют дополнительные знания о кластеризуемом множестве: распределение по кластерам, количество кластеров и т.д.
- Внутренние** (англ. *Internal*) меры отображают качество кластеризации только по информации в данных и не используя внешней информации.

Внешние меры оценки качества – Таблица сопряженности

Дано множество S из n элементов, разделение на классы $X = \{X_1, X_2, \dots, X_r\}$, и полученное разделение на кластеры $Y = \{Y_1, Y_2, \dots, Y_s\}$, совпадения между X и Y могут быть отражены в таблице сопряженности $[n_{ij}]$, где каждое n_{ij} обозначает число объектов, входящих как в X_i , так и в Y_j : $n_{ij} = |X_i \cap Y_j|$.

$X \setminus Y$	Y_1	Y_2	\dots	Y_s	Sums
X_1	n_{11}	n_{12}	\dots	n_{1s}	a_1
X_2	n_{21}	n_{22}	\dots	n_{2s}	a_2
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
X_r	n_{r1}	n_{r2}	\dots	n_{rs}	a_r
Sums	b_1	b_2	\dots	b_s	n

Пусть $p_{ij} = \frac{n_{ij}}{n}$, $p_i = \frac{a_i}{n}$, $p_j = \frac{b_j}{n}$.

Также рассмотрим пары (x_i, x_j) из элементов кластеризуемого множества X . и подсчитаем количество пар, в которых:

- Элементы принадлежат одному кластеру и одному классу — TP
- Элементы принадлежат одному кластеру, но разным классам — FP
- Элементы принадлежат разным кластерам, но одному классу — FN
- Элементы принадлежат разным кластерам и разным классам — TN

Внешние меры оценки качества - Индексы

- **Индекс Rand** $Rand = \frac{TP + TN}{TP + TN + FP + FN}$
- **Индекс Жаккара (Jaccard Index)** $Jaccard = \frac{TP}{TP + FN + FP}$
- **Индекс Фоулкса – Мэллова (Fowlkes-Mallows Index)** $FM = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}}$
- **Индекс Phi** $\Phi = \frac{TP \times TN - FN \times FP}{(TP + FN)(TP + FP)(FN + TN)(FP + TN)}$
- **Entropy (Энтропия)** $E = - \sum_i p_i (\sum_j \frac{p_{ij}}{p_i} \log(\frac{p_{ij}}{p_i}))$
- **Purity (чистота)** $P = \sum_i \max_j p_{ij}$
- **F-мера** $F = \sum_j p_j \max_i [2 \frac{p_{ij}}{p_i} \frac{p_{ij}}{p_j} / (\frac{p_{ij}}{p_i} + \frac{p_{ij}}{p_j})]$
- **Variation of Information (изменение информации)**

$$VI = - \sum_i p_i \log p_i - \sum_j p_j \log p_j - 2 \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$$

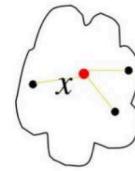
Внутренние меры оценки качества

Данные меры оценивают качество структуры кластеров опираясь только непосредственно на нее, не используя внешней информации.

- **Компактность кластеров (Cluster Cohesion)**

Within cluster Sum of Squares (WSS) → min

$$WSS = \sum_{j=1}^M \sum_{i=1}^{|C_j|} (x_{ij} - \bar{x}_j)^2, \text{ где } M \text{ — количество кластеров.}$$

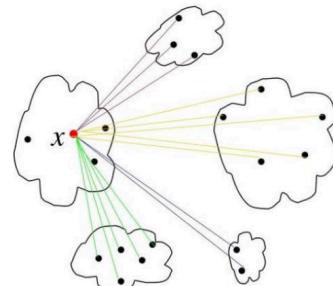


- **Отделимость кластеров (Cluster Separation)**

Between cluster Sum of Squares (BSS) → max

$$BSS = \sum_i |C_i|(m - m_i)^2$$

где C_i — i -ый кластер, m_i — центроид C_i , m — общий центр.



- **Силуэт (Silhouette)**

$$Sil(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{b(x_i, c_k) - a(x_i, c_k)}{\max\{a(x_i, c_k), b(x_i, c_k)\}},$$

$$a(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \|x_i - x_j\| \text{ - компактность.}$$

$$b(x_i, c_k) = \min_{c_l \in C \setminus c_k} \left\{ \frac{1}{|c_l|} \sum_{x_j \in c_l} \|x_i - x_j\| \right\} \text{ - отделимость.}$$

Значение: $-1 \leq Sil(C) \leq 1$. Чем ближе к 1, тем лучше.

- **Индекс Дэвиса-Болдуина (Davies–Bouldin Index)**

$$DB(C) = \frac{1}{K} \sum_{c_k \in C} \max_{c_l \in C \setminus c_k} \left\{ \frac{S(c_k) + S(c_l)}{\|\bar{c}_k - \bar{c}_l\|} \right\}, \quad S(c_k) = \frac{1}{|c_k|} \sum_{x_i \in c_k} \|x_i - \bar{c}_k\|$$

- **Индекс Calinski–Harabasz**

$$CH(C) = \frac{N - K}{K - 1} \cdot \frac{\sum_{c_k \in C} |c_k| \cdot \|\bar{c}_k - \bar{X}\|}{\sum_{c_k \in C} \sum_{x_i \in c_k} \|x_i - \bar{c}_k\|}$$

- **Score function**

$$SF(C) = 1 - \frac{1}{e^{bcd(C) - wcd(C)}},$$

$$bcd(C) = \frac{\sum_{c_k \in C} |c_k| \cdot \|\bar{c}_k - \bar{X}\|}{N \times K}, \quad wcd(C) = \sum_{c_k \in C} \frac{1}{|c_k|} \sum_{x_i \in c_k} \|x_i - \bar{c}_k\|$$

48. Кластеризация. Типы алгоритмов кластеризации.

Кластеризация (clustering/cluster analysis) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Типы алгоритмов кластеризации

Алгоритмы кластеризации можно разделить на 4 типа:

- Кластерный центроид.** Он объединяет данные в кластеры, основываясь на заранее заданных условиях и характеристиках. k-средних — наиболее популярный алгоритм из этой категории.
- Кластеризация на основе плотности.** В этом типе используется алгоритм, который соединяет области с высокой плотностью в кластеры, создавая распределения произвольной формы.
- Кластеризация на основе распределений.** Алгоритм этого типа предполагает гауссовские распределения данных, которые далее объединяются в различные варианты того же распределения.
- Иерархические алгоритмы.** В этом алгоритме строится иерархическая древо кластеров. Количество кластеров можно менять, объединяя их на определённом уровне древа.

49. Кластеризация. Метод k-средних (k-means).

Достоинства и недостатки.

Кластеризация (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Метод k-средних (k-means)

Алгоритм относится к классу эвристических EM-алгоритмов (Expectation-maximization), используемых в математической статистике для нахождения оценок максимального правдоподобия параметров вероятностных моделей.

Основная идея метода — итеративное повторение двух шагов:

- распределение объектов выборки по кластерам;
- пересчёт центров кластеров.

Алгоритм заключается в том, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{k=1}^K \sum_{x_i \in C_k} \rho^2(x_i, c_k) \rightarrow \min_c$$

где K — число кластеров, C_k — полученные кластеры, c_k — центр кластера C_k , x_i вектор из кластера C_k .

В начале работы алгоритма выбираются K случайных центров в пространстве признаков. Каждый объект выборки относят к тому кластеру, к центру которого объект оказался ближе по выбранной метрике.

$$x_i \in C_k, \text{ если } \rho(x_i, c_k) = \min_{k=1, K}$$

Далее центры кластеров пересчитывают как среднее арифметическое векторов признаков всех вошедших в этот кластер объектов (то есть центр масс кластера). Как только мы обновили центры кластеров, объекты заново перераспределяются по ним, а затем можно снова уточнить положение центров. Процесс продолжается до тех пор, пока центры кластеров не перестанут меняться.

Недостатки метода k-средних

- Не гарантирует достижения глобального минимума суммарного квадратичного отклонения V , а только одного из локальных минимумов.
- Результат зависит от начального выбора центров кластеров $\{c_k^{(0)}\}$, а их оптимальный выбор неизвестен.
- Число кластеров K надо знать заранее.

Инициализация центров кластеров

- Метод Forgy. Из имеющегося набора данных случайным образом выбираются K наблюдений.
- Случайное разбиение. Каждому наблюдению на начальном этапе случайным образом присваивается номер кластера.
- Алгоритм k-means++. Из имеющегося набора данных случайным образом выбирается одна точка (первый центроид). Затем следующая точка выбирается из оставшихся с вероятностью, пропорционально зависящей от квадрата расстояния от точки до ближайшего центроида. Итерации повторяются до тех пор, пока не будут выбраны K центроидов.

50. Кластеризация. Алгоритмы семейства FOREL. Достоинства и недостатки.

Кластеризация (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Алгоритмы семейства FOREL

FOREL (Формальный Элемент) — алгоритм кластеризации, основанный на идее объединения в один кластер объектов в областях их наибольшего сгущения.

Цель: Разбить выборку на такое (заранее неизвестное) число таксонов (групп/кластеров), чтобы сумма расстояний от объектов кластеров до центров кластеров была минимальной по всем кластерам. То есть наша задача — выделить группы максимально близких друг к другу объектов, которые в силу гипотезы схожести и будут образовывать наши кластеры.

Входные данные

- Кластеризуемая выборка - может быть задана признаковыми описаниями объектов либо матрицей попарных расстояний между объектами.
- Параметр R — радиус поиска локальных сгущений (можно задавать как из априорных соображений, так и настраивать скользящим контролем).
- В модификациях возможно введение параметра k — количества кластеров.

Выходные данные

- Кластеризация на заранее неизвестное число таксонов.

Минимизируемый алгоритмом функционал качества

$$F = \sum_{j=1}^k \sum_{x \in K_j} \rho(x, W_j),$$

где первое суммирование ведется по всем кластерам выборки, второе суммирование — по всем объектам x , принадлежащим текущему кластеру K_j , а W_j — центр текущего кластера, $\rho(x, y)$ — расстояние между объектами.

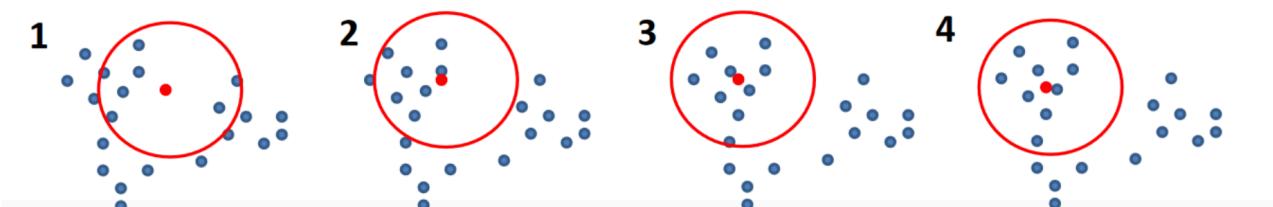
FOREL- Алгоритм

Алгоритм

1. Случайно выбираем текущий объект из выборки;
2. Помечаем объекты выборки, находящиеся на расстоянии менее, чем R от текущего;
3. Вычисляем их центр тяжести, помечаем этот центр как новый текущий объект;
4. Повторяем шаги 2-3, пока новый текущий объект не совпадет с прежним;
5. Помечаем объекты внутри сферы радиуса R вокруг текущего объекта как кластеризованные, выкидываем их из выборки;
6. Повторяем шаги 1-5, пока не будет кластеризована вся выборка.

Выбор центра тяжести:

- В линейном пространстве — центр масс;
- В метрическом пространстве — объект, сумма расстояний до которого минимальна, среди всех внутри сферы;
- Объект, который внутри сферы радиуса R содержит максимальное количество других объектов из всей выборки (медленно);
- Объект, который внутри сферы маленького радиуса содержит максимальное количество объектов (из сферы радиуса R).



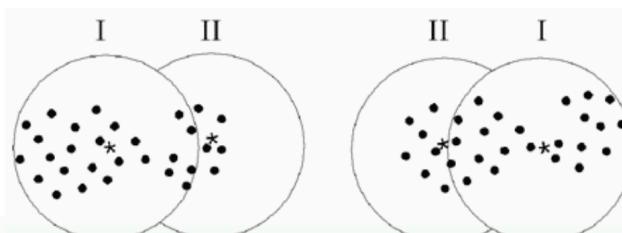
FOREL- Преимущества и Недостатки

Преимущества

- Точность минимизации функционала качества (при удачном подборе параметра R);
- Наглядность визуализации кластеризации;
- Сходимость алгоритма;
- Возможность операций над центрами кластеров — они известны в процессе работы алгоритма;
- Возможность подсчета промежуточных функционалов качества, например, длины цепочки локальных сгущений;
- Возможность проверки гипотез схожести и компактности в процессе работы алгоритма.

Недостатки

- Относительно низкая производительность (решается введением функции пересчета поиска центра при добавлении 1 объекта внутрь сферы);
- Плохая применимость алгоритма при плохой разделимости выборки на кластеры;
- Неустойчивость алгоритма (зависимость от выбора начального объекта);
- Произвольное по количеству разбиение на кластеры;
- Необходимость априорных знаний о ширине (диаметре) кластеров.



51. Кластеризация. Алгоритм DBSCAN. Достоинства и недостатки.

Кластеризация (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) - Основанная на плотности пространственная кластеризация для приложений с шумами.

Если дан набор точек в некотором пространстве, алгоритм группирует вместе точки, которые тесно расположены (точки со многими близкими соседями), помечая как выбросы точки, которые находятся одиноко в областях с малой плотностью (ближайшие соседи которых лежат далеко). **DBSCAN** развивает идею кластеризации с помощью выделения связных компонент.

Плотность в DBSCAN определяется в окрестности каждого объекта выборки x_i как количество других точек выборки в шаре $B(\varepsilon, x_i)$. Кроме радиуса ε окрестности в качестве гиперпараметра алгоритма задается порог **MinPts** по количеству точек в окрестности.

Все объекты выборки делятся на три типа:

- **внутренние / основные точки (core points)**,
- **границные/ достижимые по плотности (border points)**
- **шумовые/ выпадающие точки (noise points)**.

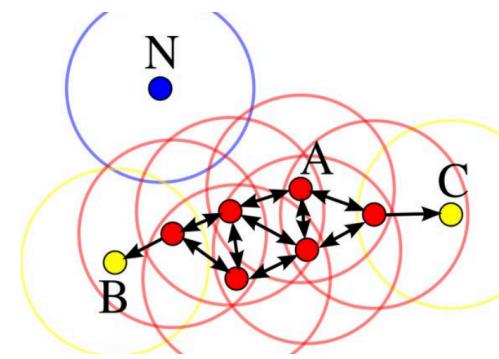
К основным относятся точки, в окрестности которых больше *MinPts* объектов выборки. К границным — точки, в окрестности которых есть основные, но общее количество точек в окрестности меньше *MinPts*. Шумовыми называют точки, в окрестности которых нет основных точек и в целом содержится менее *MinPts* объектов выборки.

DBSCAN - Алгоритм

1. Выделяются основные точки.
2. Основные точки, у которых есть общая окрестность, соединяются ребром. Говорят, что эти точки *прямо достижимы*.
3. В полученном графе выделяются компоненты связности.
4. Каждая граничная точка (*достижимая точка*) относится к тому кластеру, в который попала ближайшая к ней основная точка.
5. Шумовые точки (не достижимые из основных точек) убираются из рассмотрения и не приписываются ни к какому кластеру.

Если A является основной точкой, то она формирует *кластер* вместе со всеми точками (основными или неосновными), достижимые из этой точки. Каждый кластер содержит по меньшей мере одну основную точку. Неосновные точки могут быть частью кластера, но они формируют его «край», поскольку не могут быть использованы для достижения других точек.

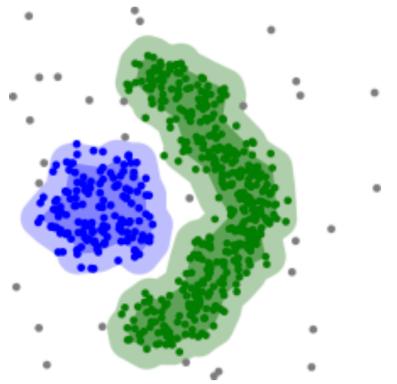
- Две точки P и Q связаны по плотности, если имеется точка O , такая что и P , и Q достижимы из O . Кластер удовлетворяет двум свойствам:
- Все точки в кластере попарно связны по плотности.
 - Если точка достижима по плотности из какой-то точки кластера, она также принадлежит кластеру.



DBSCAN - Преимущества и Недостатки

Преимущества

- Не требует спецификации числа кластеров в данных.
- Может найти кластеры произвольной формы.
- Имеет понятие шума и устойчив к выбросам.
- Требует лишь двух параметров и большей частью нечувствителен к порядку выбора точек.
- Параметры minPts и ε могут быть установлены экспертами в рассматриваемой области, если данные хорошо понимаются.



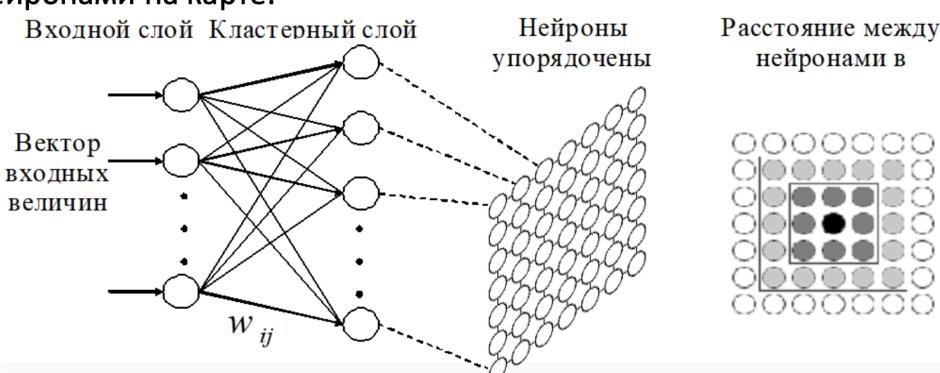
Недостатки

- Не полностью однозначен — краевые точки, которые могут быть достигнуты из более чем одного кластера, могут принадлежать любому из этих кластеров, что зависит от порядка просмотра точек.
- Качество DBSCAN зависит от измерения расстояния, используемого для подсчета количества других точек выборки в сфере $B(\varepsilon, x_i)$.
- Не может хорошо кластеризовать наборы данных с большой разницей в плотности, поскольку не удается выбрать приемлемую для всех кластеров комбинацию minPts и ε .
- Если есть сложности с пониманием особенностей данных и масштаба, выбор порога расстояния ε может оказаться проблематичным.

52. Самоорганизующиеся карты Кохонена. Решаемые задачи. Обучение сети.

Самоорганизующиеся карты Кохонена (SOM)

Состоит из входного слоя и одного конкурирующего кластерного слоя, нейроны которого вычисляют расстояние между входным вектором и вектором весовых коэффициентов. Победителем является тот нейрон, расстояние до которого минимально. Важная особенность алгоритма SOM (Self Organizing Maps) заключается в том, что все нейроны кластерного слоя (ядра классов) упорядочены в некоторую структуру, что позволяет ввести меру взаимодействия между нейронами кластерного слоя не в пространстве входных признаков, а на используемой карте. Величина этого взаимодействия определяется расстоянием между нейронами на карте.



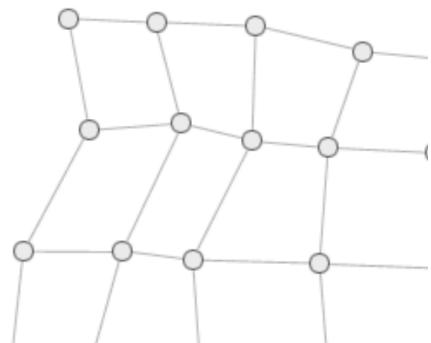
SOM - Обучение сети

- инициализацию сети (количество нейронов в сети и условие взаимодействия между нейронами кластерного слоя)
- инициализация весовых коэффициентов
 - Инициализация случайными значениями
 - Инициализация примерами
 - Линейная инициализация.
- Обучение (корректировка весов). Подстройке подлежат веса, как нейрона-победителя, так и его соседей по сетке (одно или двумерной), но в меньшей степени.

$$w_i(t+1) = w_i(t) + h_{ci}(t) * [x(t) - w(t)]$$

$$h(t) = h(\|a_c - a_i\|, t) * f(t)$$

$$h(d, t) = e^{-\left(\frac{d}{b(t)}\right)^2}$$



SOM - Раскраска обученной карты

Добавление дополнительных связей в пространстве отображения результата позволяют использовать сеть Кохонена, как для решения задач распознавания и определения близости кластеров в данных, так и для визуализации многомерного пространства

- в соответствии с расстояниями между векторами весовых коэффициентов нейронов кластерного слоя;
- в зависимости от значений некоторой компоненты вектора обучающей выборки;
- в зависимости от числа примеров отнесенных к соответствующему нейрону карты.

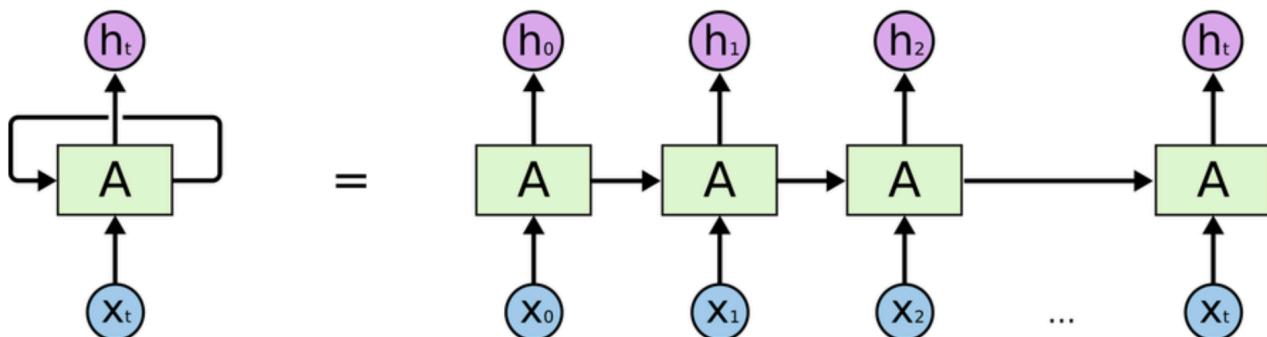
В совокупности полученные раскраски образуют атлас, отображающий присутствующие в данных кластеры близких объектов, зависимости от значений отдельных или наборов компонент, относительное расположение их различных значений, что в значительной степени может помочь пользователю разобраться в структуре обрабатываемых данных.

53. Рекуррентные нейронные сети. Принципы функционирования на примере нейронная сеть Хопфилда.

Рекуррентные нейронные сети

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) - это сети, содержащие обратные связи, позволяющие сохранять информацию (скрытое состояние) от предыдущих входных данных. Хорошо подходят для обработки последовательностей, например, временные ряды (изменения цен акций, показания датчиков), последовательности с зависимыми элементами (предложения естественного языка), т.е. любые данные, где соседние экземпляры (точки выборки) зависят друг от друга и эту зависимость нельзя игнорировать.

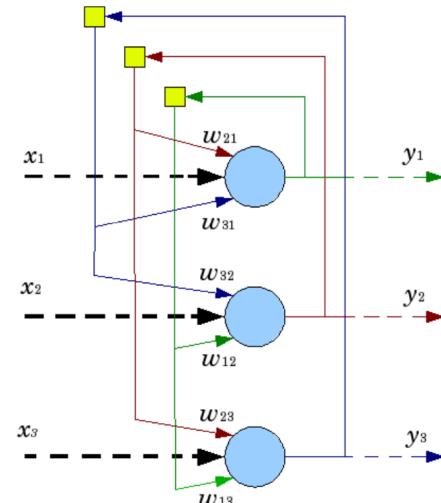
Фрагмент нейронной сети A принимает входное значение x_t и возвращает значение h_t . Наличие обратной связи позволяет передавать информацию от одного шага сети к другому. Рекуррентную сеть можно рассматривать, как несколько копий одной и той же сети, каждая из которых передает информацию последующей копии.



Нейронная сеть Хопфилда

Нейронная сеть Хопфилда - полносвязная однослойная нейронная сеть с симметричной матрицей связей. Функционирование до достижения равновесия, т.е. когда следующее состояние сети в точности равно предыдущему: начальное состояние является входным образом, а при равновесии получают выходной образ. В искусственных нейронов. Каждый нейрон системы может принимать на входе и на выходе одно из двух состояний (что аналогично выходу нейрона с пороговой функцией активации):

$$y_i(t) \in \{-1; +1\}$$



Каждый нейрон связан со всеми остальными нейронами. Взаимодействие нейронов сети описывается выражением:

$$E = \frac{1}{2} \sum_{i,j=1}^N w_{ij} x_i x_j$$

где w_{ij} – элемент матрицы взаимодействий W , которая состоит из весовых коэффициентов связей между нейронами.

Нейронная сеть Хопфилда - Состояние сети

Состояние сети – это просто множество текущих значений сигналов y_i от всех нейронов. В первоначальной сети Хопфилда состояние каждого нейрона менялось в дискретные случайные моменты времени, в последующей работе состояния нейронов могли меняться одновременно. Так как выходом бинарного нейрона может быть только два значения (промежуточных уровней нет), то текущее состояние сети является двоичным числом, каждый бит которого является сигналом y_i некоторого нейрона.

Нейронная сеть Хопфилда - Обучение

В процессе обучения формируется выходная матрица W , которая запоминает m эталонных «образов» — N -мерных бинарных векторов: $X_m = (x_{m1}, x_{m2}, \dots, x_{mN})$, эти образы во время эксплуатации сети будут выражать отклик системы на входные сигналы, или иначе – окончательные значения выходов y_i после серии итераций.

Вычисление коэффициентов основано на следующем правиле: для всех запомненных образов X_i матрица связи должна удовлетворять уравнению :

$$X_i = WX_i$$

Расчёт весовых коэффициентов проводится по следующей формуле:

$$w_{ij} = \frac{1}{N} \sum_{d=1..m} X_{id} X_{jd}$$

где N – размерность векторов, m – число запоминаемых выходных векторов, d – номер запоминаемого выходного вектора, X_{ij} – i -я компонента запоминаемого выходного j -го вектора.

Может быть записано в векторном виде :

$$W = \frac{1}{N} \sum_i X_i X_i^T$$

где X_i – i -й запоминаемый вектор-столбец.

В работе Кохонена и Гроссберга (доказана теорема) показано, что сеть с обратными связями является устойчивой, если её матрица симметрична и имеет нули на главной диагонали.

Применение обученной сети

Обученная сеть способна распознавать входные сигналы – то есть, определять, к какому из запомненных образцов они относятся. Сеть последовательно меняет свои состояния согласно формуле:

$$X(t + 1) = F(W \cdot X(t))$$

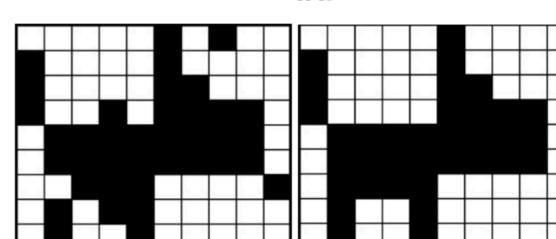
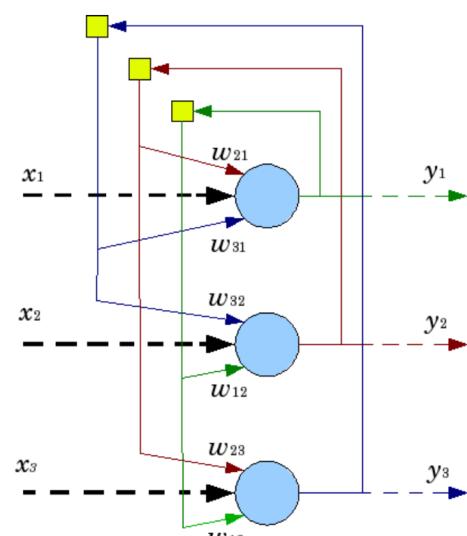
где F – активационная функция, $X(t)$ и $X(t + 1)$ – текущее и следующее состояния сети.

Локальным полем a_i , действующим на нейрон x_i со стороны всех остальных нейронов сети, является значение:

$$a_i(t) = \sum_{j=1, j \neq i}^N w_{ji} x_j(t - 1)$$

Значение выхода нейрона i в текущий момент времени $x_i(t)$ рассчитывается по формуле:

$$x_i(t) = \text{sign} \left(\sum_{j=1, j \neq i}^N w_{ji} x_j(t - 1) \right)$$



Режимы работы

Две модификации (режима работы), отличающиеся по времени передачи сигнала: **Синхронный** и **Асинхронный**.

Синхронный режим работы сети - последовательно просматриваются нейроны, их состояния запоминаются отдельно и не меняются до тех пор, пока не будут пройдены все нейроны сети. Когда все нейроны просмотрены, их состояния одновременно (то есть синхронно) меняются на новые.

Асинхронный режим работы сети – состояния нейронов в следующий момент времени меняются последовательно: вычисляется локальное поле для первого нейрона в момент t , определяется его реакция, и нейрон устанавливается в новое состояние (которое соответствует его выходу в момент $t + 1$), и так далее – состояние каждого следующего нейрона вычисляется с учетом всех изменений состояний рассмотренных ранее нейронов.

Ограничения сети

1. Относительно небольшой объём памяти, величину которого можно оценить выражением:

$$M \approx \frac{N}{2 \cdot \log_2 N}$$

Попытка записи большего числа образов приводит к тому, что нейронная сеть перестаёт их распознавать.

2. В синхронном режиме сеть может прийти к динамическому аттрактору.

3. Достижение устойчивого состояния не гарантирует правильный ответ сети. Это происходит из-за того, что сеть может сойтись к так называемым ложным аттракторам, иногда называемым «химерами» (как правило, химеры склеены из фрагментов различных образов возникают при слишком большом количестве запомненных образов).

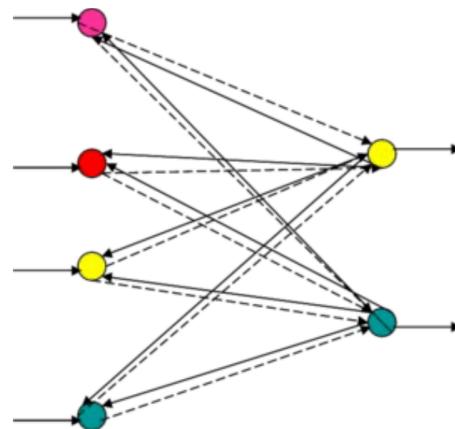
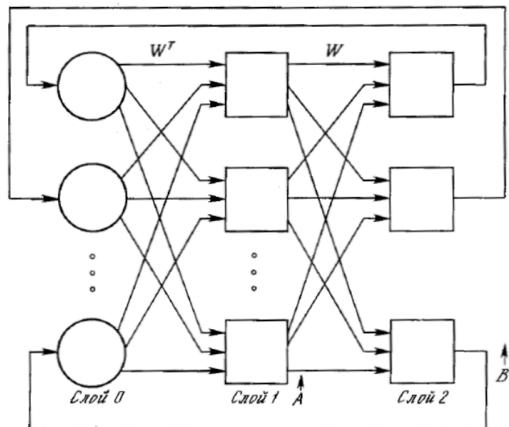
54. Рекуррентные нейронные сети. Двунаправленная ассоциативная память.

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) - это сети, содержащие обратные связи, позволяющие сохранять информацию (скрытое состояние) от предыдущих входных данных. Хорошо подходят для обработки последовательностей, например, временные ряды (изменения цен акций, показания датчиков), последовательности с зависимыми элементами (предложения естественного языка), т.е. любые данные, где соседние экземпляры (точки выборки) зависят друг от друга и эту зависимость нельзя игнорировать.

Фрагмент нейронной сети А принимает входное значение x_t и возвращает значение h_t . Наличие обратной связи позволяет передавать информацию от одного шага сети к другому. Рекуррентную сеть можно рассматривать, как несколько копий одной и той же сети, каждая из которых передает информацию последующей копии.

Двунаправленная ассоциативная память (ДАП) является гетероассоциативной; входной вектор поступает на один набор нейронов, а соответствующий выходной вектор вырабатывается на другом наборе нейронов.

На рисунке приведены две базовые конфигурации ДАП. Одна из них выбрана таким образом, чтобы подчеркнуть сходство с сетями Хопфилда и предусмотреть увеличения количества слоев.



ДАП – работа сети

Входной вектор A обрабатывается матрицей весов W сети, в результате чего вырабатывается вектор выходных сигналов нейронов B . Вектор B затем обрабатывается транспонированной матрицей W^T весов сети, которая вырабатывает новые выходные сигналы, представляющие собой новый входной вектор A . Этот процесс повторяется до тех пор, пока сеть не достигнет стабильного состояния, в котором ни вектор A , ни вектор B не изменяются.

В векторной форме:

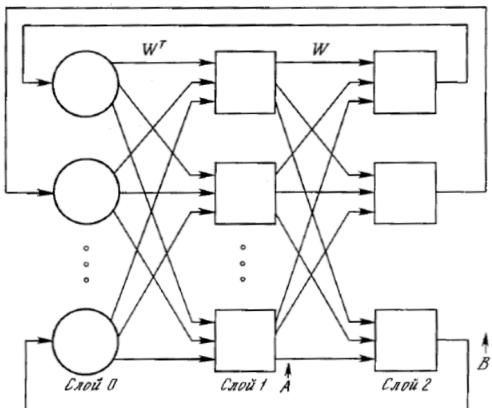
$$B = F(AW)$$

где B – вектор выходного сигнала нейронов слоя 2, A – вектор выходных сигналов нейрона слоя 1, W – матрица весов связей между слоями 1 и 2, F – функция активации.

$$A = F(BW^T)$$

где W^T – является транспонированной матрицей W .

Сеть функционирует в направлении минимизации функции энергии Ляпунова в основном таким же образом, как и сети Хопфилда в процессе сходимости. Таким образом, каждый цикл модифицирует систему в направлении энергетического минимума, расположение которого определяется значениями весов.



ДАП – Кодирование ассоциаций и Емкость памяти

Обучение производится с использованием обучающего набора, состоящего из пар векторов A и B . Процесс обучения реализуется в форме вычислений; это означает, что весовая матрица вычисляется как сумма произведений всех векторных пар обучающего набора. В символьной форме:

$$W = \sum_i A_i^T B_i$$

Существует взаимосвязь между ДАП и рассмотренными сетями Хопфилда. Если весовая матрица W является квадратной и симметричной, то $W = W^T$. В этом случае, если слои 1 и 2 являются одним и тем же набором нейронов, ДАП превращается в автоассоциативную сеть Хопфилда.

Емкость сети. Если M векторов выбраны случайно и представлены в указанной выше форме, и если M меньше чем:

$$M_1 \approx \frac{N}{2 \cdot \log_2 N}$$

где N – количество нейронов в наименьшем слое, тогда все запомненные образы, за исключением «малой части», могут быть восстановлены.

Если все образы должны восстанавливаться, M должно быть меньше

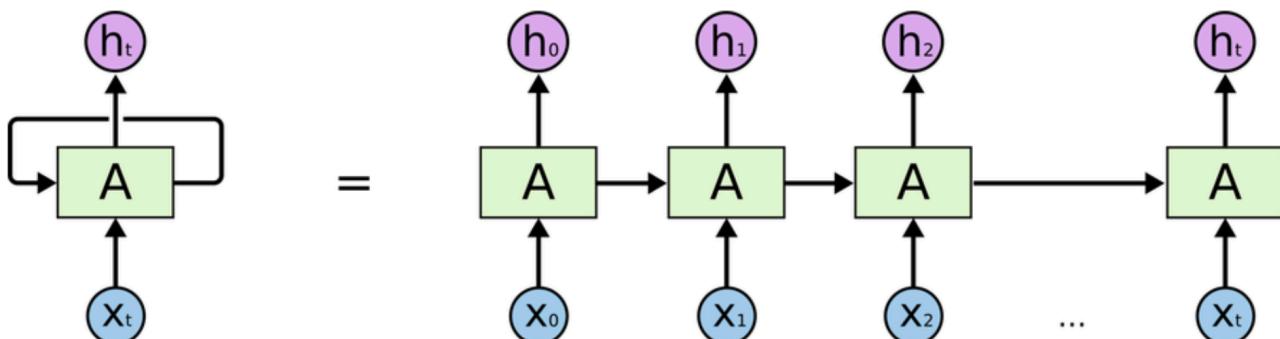
$$M_2 \approx \frac{N}{4 \cdot \log_2 N}$$

Например, если $N = 1024$, тогда M_1 должно быть меньше 51, а M_2 меньше 25.

55. Рекуррентные нейронные сети. Проблема долговременных зависимостей.

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) - это сети, содержащие обратные связи, позволяющие сохранять информацию (скрытое состояние) от предыдущих входных данных. Хорошо подходят для обработки последовательностей, например, временные ряды (изменения цен акций, показания датчиков), последовательности с зависимыми элементами (предложения естественного языка), т.е. любые данные, где соседние экземпляры (точки выборки) зависят друг от друга и эту зависимость нельзя игнорировать.

Фрагмент нейронной сети A принимает входное значение x_t и возвращает значение h_t . Наличие обратной связи позволяет передавать информацию от одного шага сети к другому. Рекуррентную сеть можно рассматривать, как несколько копий одной и той же сети, каждая из которых передает информацию последующей копии.



Одна из привлекательных идей RNN состоит в том, что они потенциально умеют связывать предыдущую информацию с текущей задачей, так, например, знания о предыдущем кадре видео могут помочь в понимании текущего кадра. Если бы RNN обладали такой способностью, они были бы чрезвычайно полезны. Но действительно ли RNN предоставляют нам такую возможность? Это зависит от некоторых обстоятельств.

Иногда для выполнения текущей задачи нам необходима только недавняя информация. Рассмотрим, например, языковую модель, пытающуюся предсказать следующее слово на основании предыдущих. Если мы хотим предсказать последнее слово в предложении "облака плывут по небу", нам не нужен более широкий контекст; в этом случае довольно очевидно, что последним словом будет "небу". В этом случае, когда дистанция между актуальной информацией и местом, где она понадобилась, невелика, RNN могут обучиться использованию информации из прошлого.

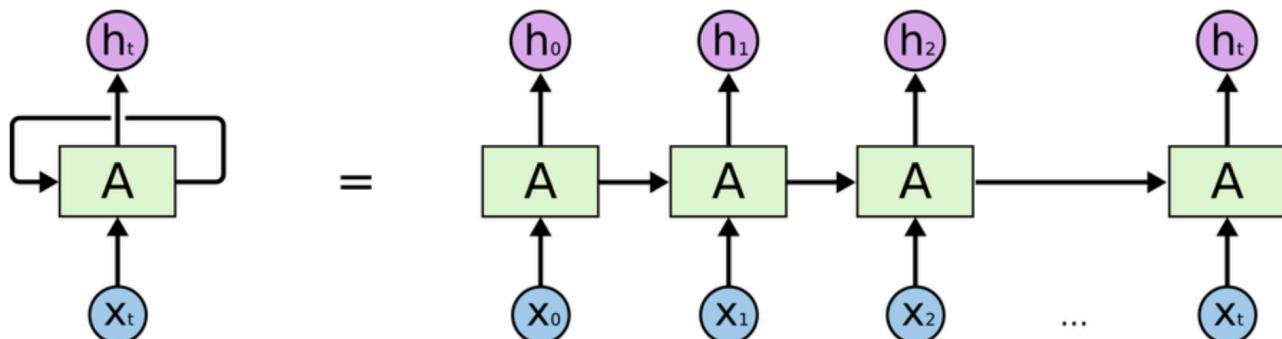
Но бывают случаи, когда нам необходимо больше контекста. Допустим, мы хотим предсказать последнее слово в тексте "Я вырос во Франции... Я бегло говорю по-французски". Ближайший контекст предполагает, что последним словом будет называние языка, но чтобы установить, какого именно языка, нам нужен контекст Франции из более отдаленного прошлого. Таким образом, разрыв между актуальной информацией и точкой ее применения может стать очень большим.

К сожалению, по мере роста этого расстояния, RNN теряют способность связывать информацию.

56. Рекуррентные нейронные сети. Долгая краткосрочная память - Long short-term memory (LSTM)

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) - это сети, содержащие обратные связи, позволяющие сохранять информацию (скрытое состояние) от предыдущих входных данных. Хорошо подходят для обработки последовательностей, например, временные ряды (изменения цен акций, показания датчиков), последовательности с зависимыми элементами (предложения естественного языка), т.е. любые данные, где соседние экземпляры (точки выборки) зависят друг от друга и эту зависимость нельзя игнорировать.

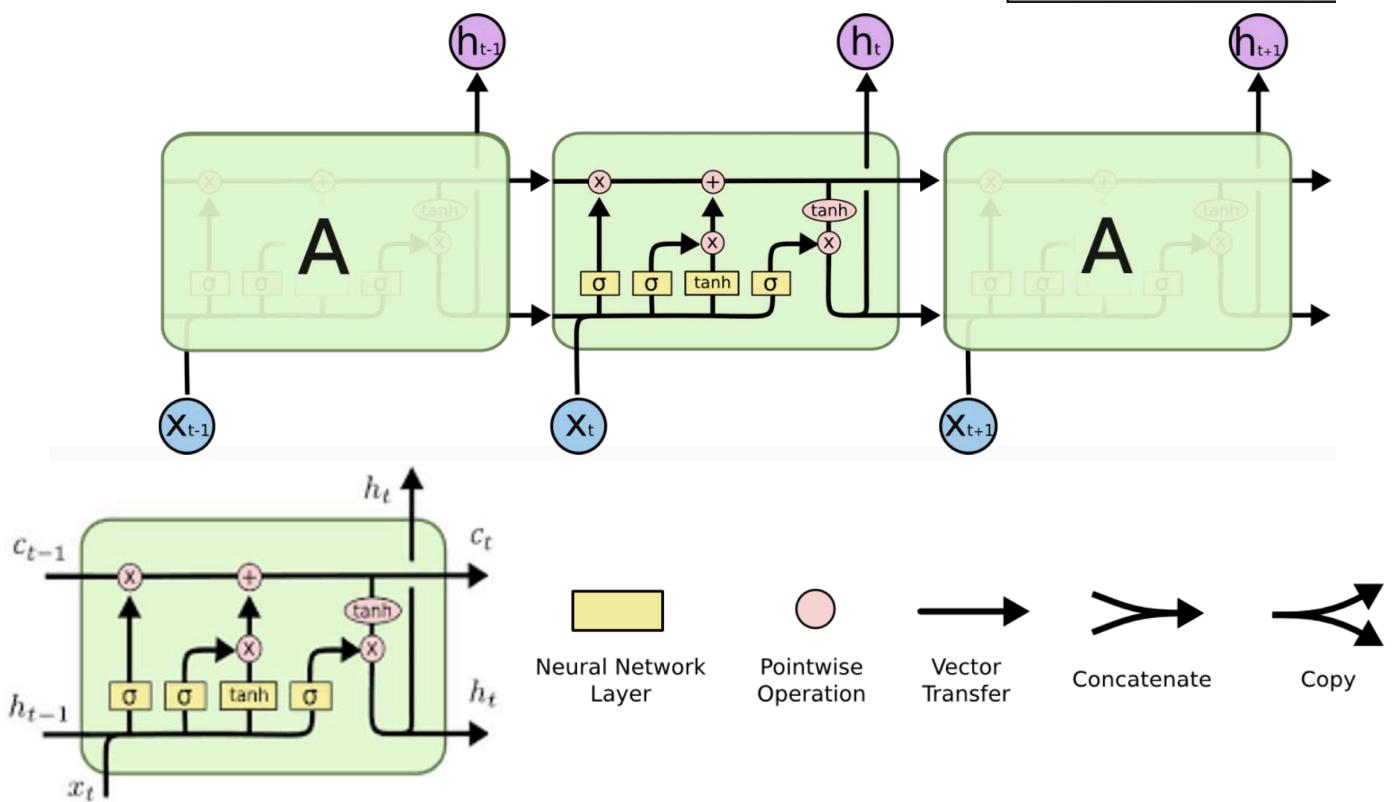
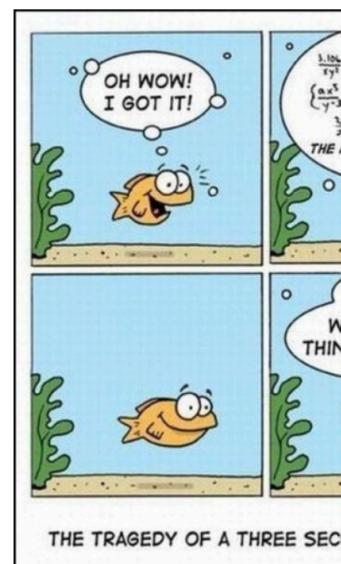
Фрагмент нейронной сети A принимает входное значение x_t и возвращает значение h_t . Наличие обратной связи позволяет передавать информацию от одного шага сети к другому. Рекуррентную сеть можно рассматривать, как несколько копий одной и той же сети, каждая из которых передает информацию последующей копии.



Долгая краткосрочная память – Long short-term memory (LSTM)

Особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям, предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером.

LSTM-модули разработаны специально, чтобы избежать проблемы долговременной зависимости, запоминая значения как на короткие, так и на длинные промежутки времени. Хранимое значение не размывается во времени и градиент не исчезает при обучении с использованием метода обратного распространения ошибки во времени (Backpropagation Through Time - BPTT).



- Слой нейронной сети (Желтый прямоугольник);
- Поточечная операция (Розовая окружность) - например, сложение векторов;
- Векторный перенос - каждая линия переносит целый вектор от выхода одного узла ко входу другого;
- Объединение (Сливающиеся линии);
- Копирование (разветвляющиеся стрелки) - данные копируются и копии уходят в разные компоненты сети.

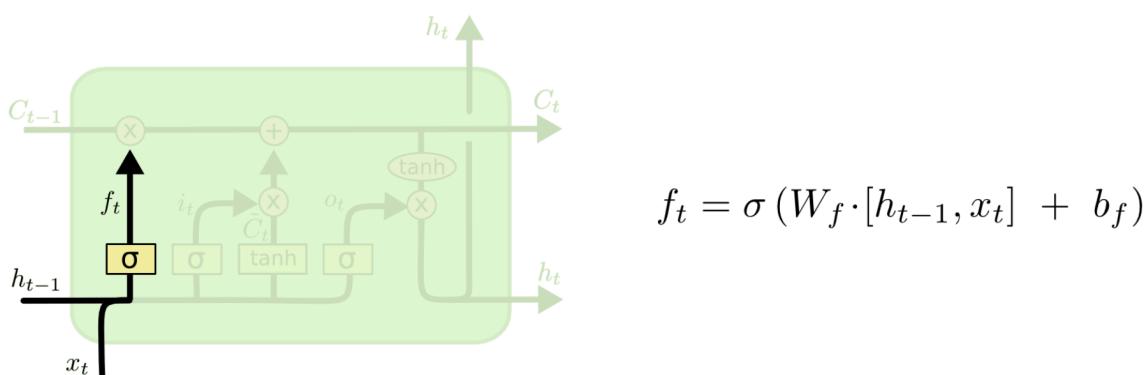
Ключевые компоненты LSTM-модуля

- **Состояние ячейки** (cell state) - память сети, которая передает соответствующую информацию по всей цепочке модулей.
- **Фильтры**, контролирующие состояние ячейки - контролируют поток информации на входах и на выходах модуля на основании некоторых условий. Состоит из слоя сигмоидальной нейронной сети и операции поточечного умножения.
 - Забывания - контролирует меру сохранения значения в памяти;
 - Входной - контролирует меру вхождения нового значения в память.
 - Выходной - контролирует меру того, в какой степени значение, находящееся в памяти, используется при расчёте выходной функции активации.

Принцип работы LSTM-модуля – Шаг 1

Определить, какую информацию можно выбросить из состояния ячейки. Для этого используется «**слой фильтра забывания**» (англ. *forget gate layer*).

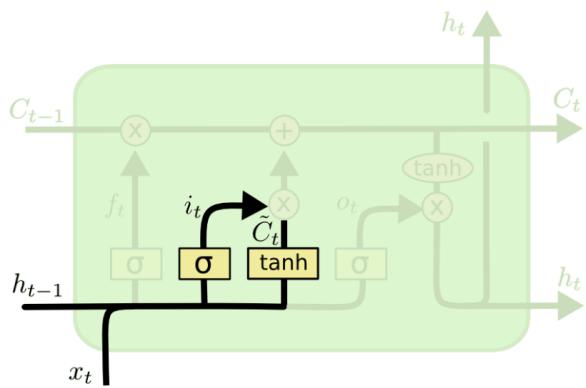
Значения предыдущего выхода h_{t-1} и текущего входа x_t пропускаются через сигмоидальный слой. Полученные значения находятся в диапазоне [0; 1]. Значения, которые ближе к 0 будут забыты, а к 1 оставлены.



Принцип работы LSTM-модуля – Шаг 2

Далее решается, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей.

- Сначала сигмоидальный слой под названием “**слой входного фильтра**” (англ. *input layer gate*) определяет, какие значения следует обновить.
- Затем \tanh -слой строит вектор новых значений-кандидатов \tilde{C}_t , которые можно добавить в состояние ячейки.

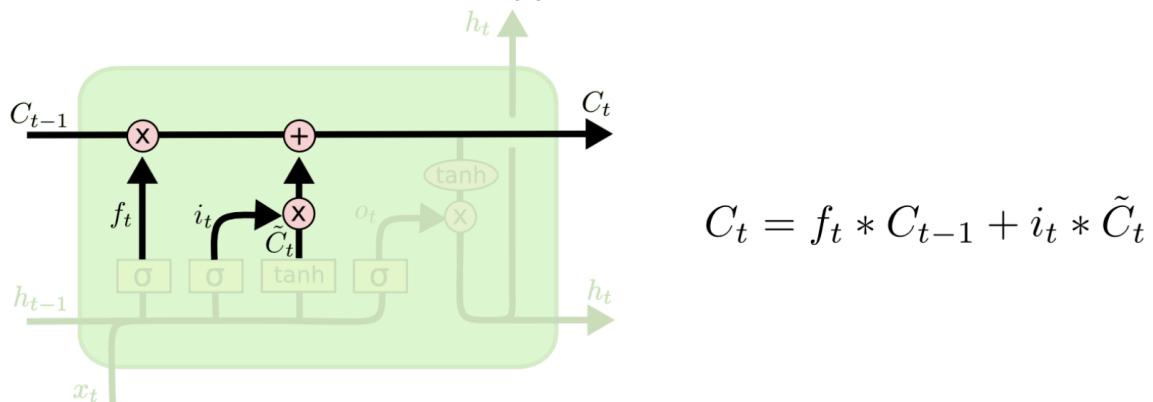


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Принцип работы LSTM-модуля – Шаг 3

Для замены старого состояния ячейки C_{t-1} на новое состояние C_t . Необходимо умножить старое состояние на f_t , забывая то, что решили забыть ранее. Затем прибавляем $i_t * \tilde{C}_t$. Это новые значения-кандидаты, умноженные на i_t – на сколько обновить каждое из значений состояния.

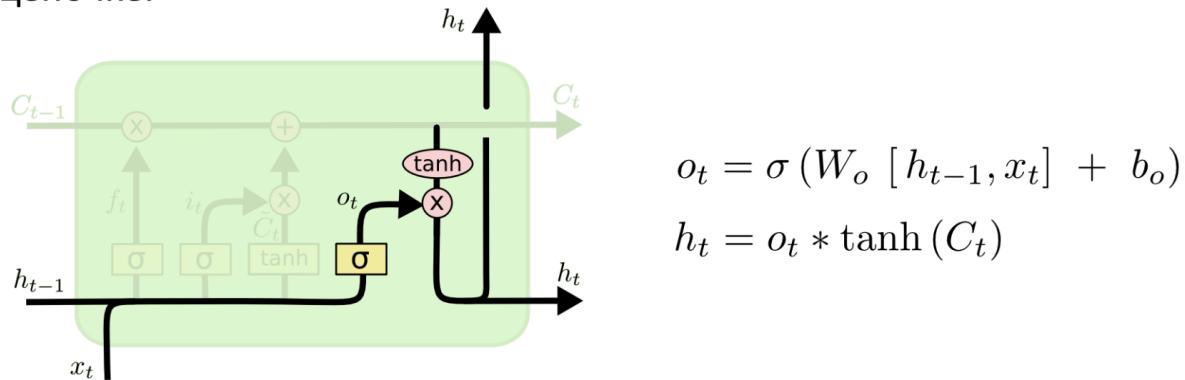


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Принцип работы LSTM-модуля – Шаг 4

На последнем этапе определяется то, какая информация будет получена на выходе. Выходные данные будут основаны на нашем состоянии ячейки, к ним будут применены некоторые фильтры. Сначала значения предыдущего выхода h_{t-1} и текущего входа x_t пропускаются через сигмоидальный слой, который решает, какая информация из состояния ячейки будет выведена. Затем значения состояния ячейки проходят через \tanh -слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию.

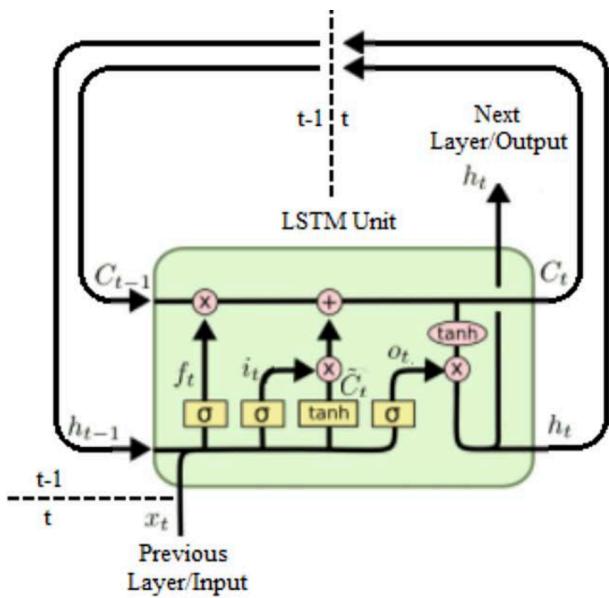
Полученные таким образом h_t и C_t передаются далее по цепочке.



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Работа LSTM-модуля



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

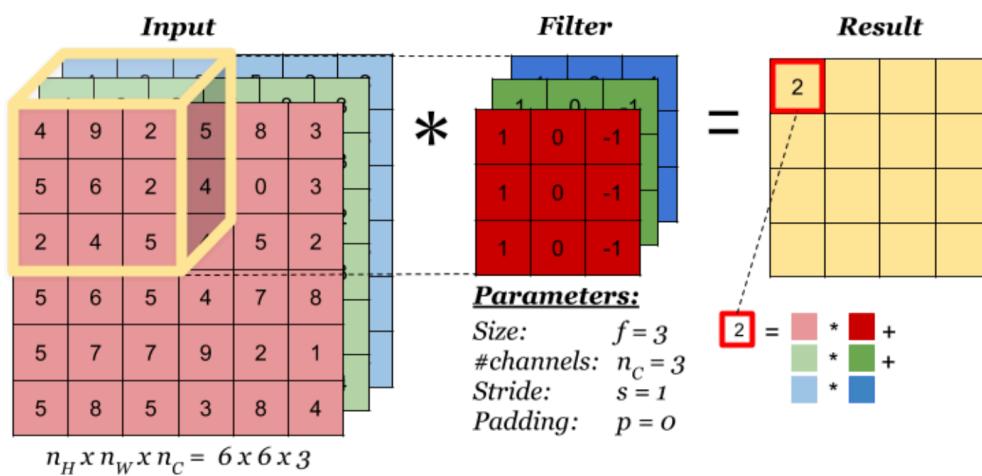
57. Сверточные нейронные сети. Принципы функционирования применительно к обработке изображений.

- Неокогнитрон (*Neocognitron*) — это иерархическая многослойная нейронная сеть сверточного типа, предложена Кунихико Фукусимой в 1980 году, способная к робастному распознаванию образов, обучаемая по принципу «обучение без учителя».
- В 1998 году Яну Лекуну удалось использовать алгоритм обратного распространения ошибки для обучения глубокой сверточной нейронной сети для решения задачи распознавания рукописных ZIP-кодов.

Операция свертки (Convolution)

Операция свертки (Convolution) аналогична использованию малого фильтра, например, размером 3×3 с шагом 1 ко всему изображению. Применение одного такого фильтра фактически является построением некой карты нахождения определенного признака на изображении. Каждый фильтр соответствует одному нейрону.

Свертка - операция над парой матриц A (размера $n_x \times n_y$) и B (размера $m_x \times m_y$), результатом которой является матрица $C = A * B$ размера $(n_x - m_x + 1) \times (n_y - m_y + 1)$. Каждый элемент результата вычисляется как скалярное произведение матрицы B и некоторой подматрицы A такого же размера (подматрица определяется положением элемента в результате). То есть, $C_{i,j} = \sum_{u=0}^{m_x-1} \sum_{v=0}^{m_y-1} A_{i+u, j+v} B_{u,v}$.



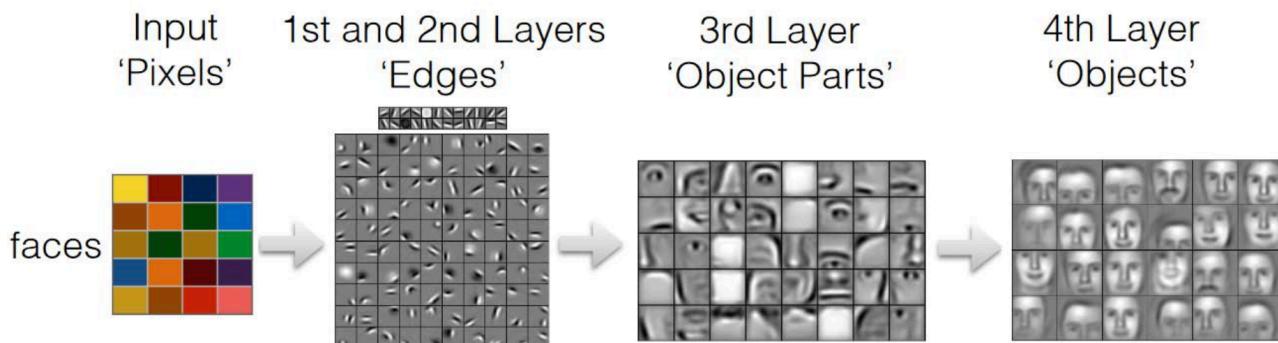
Пулинг / сабсемплинг (Subsampling)

Пулинг (или сабсемплинг) используется для уменьшения размерности. Он основан на том факте, что изображения обладают свойством локальной коррелированности пикселей, т.е. соседние пиксели, как правило, не сильно отличаются друг от друга. Таким образом, если из нескольких соседних пикселей получить какой-либо агрегат, то потери информации будут незначительными. Рекомендуемый размер объединения составляет 2×2 .

Общий принцип функционирования

На вход сети подается изображение и к нему последовательно применяются операции свертки (Convolution), которые чередуются с пулингом (Subsampling) несколько раз, а затем полученные данные проходят через набор полносвязных слоев.

Тем самым сверточные нейронные сети, позволяют создавать модели, состоящие из множества слоев, которые способны обучаться представлениям данных с различными уровнями абстракции.



58. Ключевые этапы реализации глубоких сетей.

Глубокое обучение — это совокупность методов машинного обучения, основанных на обучении представлениям (*feature/representation learning*), а не специализированных алгоритмах под конкретные задачи, и которые:

- Используют многослойную систему нелинейных фильтров для извлечения признаков с преобразованиями. Каждый последующий слой получает на входе выходные данные предыдущего слоя;
- Могут сочетать алгоритмы обучения с учителем, с частичным привлечением учителя, без учителя, с подкреплением;
- Формируют в процессе обучения слои выявления признаков (*features*) на нескольких уровнях представлений, которые соответствуют различным уровням абстракции; при этом признаки организованы иерархически — признаки более высокого уровня являются производными от признаков более низкого уровня.

Как правило, глубокое обучение предназначено для работы с большими объемами данных и использует сложные алгоритмы для обучения модели.

Нейронные сети с числом скрытых слоев большим единицы называются глубокими. Они могут содержать меньшее число нейронов в каждом слое, чем сети с одним скрытым слоем, реализующие то же самое отображение, однако строгой методики сопоставления таких сетей не существует.

Ключевые этапы реализации глубоких сетей

Можно выделить восемь ключевых этапов для реализации глубоких сетей:

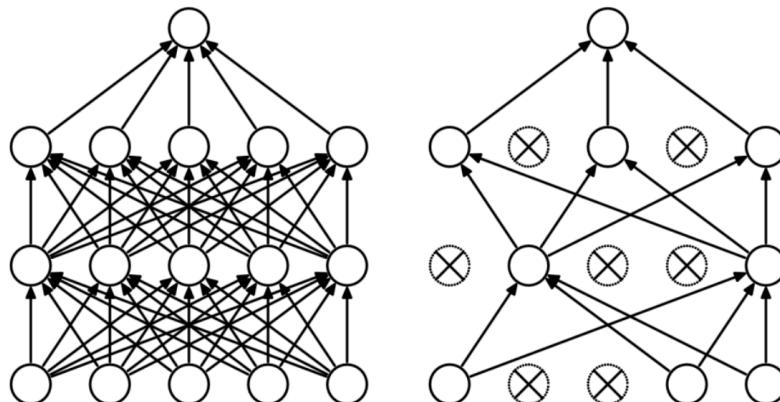
- Аугментация данных.
- Предобработка данных.
- Инициализация.
- Выбор функций активации.
- Процесс обучения.
- Регуляризация.
- Визуализация.
- Ансамбли глубоких сетей.

Аугментация данных – это методика создания дополнительных обучающих данных из уже имеющихся.

Регуляризация позволяет осуществлять контроль емкости нейронной сети, что способствует предотвращению переобучения. Основная идея заключается в учете дополнительной информации, которая имеет вид штрафа за сложность модели. Также регуляризация позволяет ограничивать значения весовых коэффициентов и осуществлять отбор наиболее важных факторов, которые сильнее всего влияют на результат.

Основные методы регуляризации применительно к нейронным сетям: L1 и L2-регуляризация, ограничения нормы вектора весов, дропаут.

Dropout регуляризация нейронной сети:



59. Подходы к извлечению правил из обученных нейронных сетей в задачах классификации.

Искусственная нейронная сеть Artificial neural network (ANN)

Математическая модель, а также ее программные или аппаратные реализации, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма.

Под искусственной нейронной сетью в дальнейшем будем понимать сеть искусственных нейронов, соединенных между собой. Здесь предполагается, что нейроны могут соединяться между собой произвольным образом и образовывать таким образом разнообразные нейронные структуры.

Извлечение правил из полносвязной нейронной сети в задачах классификации

Под извлекаемой логической закономерностью будем понимать легко интерпретируемое правило, выделяющее из обучающей выборки достаточно много объектов какого-то одного класса и практически не выделяющее объекты остальных классов. Правила, выражающие закономерности, формулируются на языке логических предикатов первого порядка вида:

ЕСЛИ (условие1) И (условие2) И ... И (условиеN) ТО (вывод).

Можно выделить следующие основные подходы:

- извлечение локальных правил из совокупности простейших однослойных сетей, на которые разделяетсястроенная многослойная модель;
- построение глобальных правил, которые характеризуют классы на выходе непосредственно через значения входных параметров.

Извлечение локальных правил из обученных нейронных сетей

Основные этапы метода NeuroRule:

Этап 1. Обучение нейронной сети.

Этап 2. Прореживание нейронной сети.

Этап 3. Подготовка к извлечению правил,
кодирование признаков классифицируемых объектов.

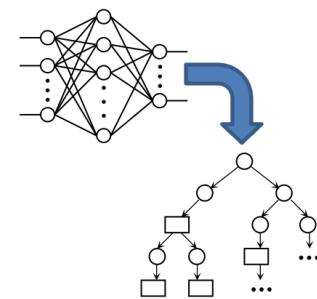
Этап 4. Извлечение правил.

Основным недостатком является наличие жестких ограничений на архитектуру нейросети, число элементов, связей и вид функций активации, т.е. отсутствие универсальности и масштабируемости.

Извлечение глобальных правил из обученных нейронных сетей

Построение глобальных правил, которые характеризуют классы на выходе непосредственно через значения входных параметров.

Данный подход осуществляет построение дерева решений на основе знаний, заложенных в обученную нейросеть, причем достаточно того, что сеть является неким «черным ящиком» или «Оракулом/Экспертом», которому можно задавать вопросы и получать от него ответы.



- Достоинство заключается в обобщающей способности искусственных нейронных сетей, что позволяет получать более простые деревья решений. Использование такого «Эксперта» позволяет компенсировать недостаток данных, наблюдающийся при построении деревьев решений на низких уровнях.
- Недостаток заключается в отсутствии прозрачности внутреннего функционирования сети.

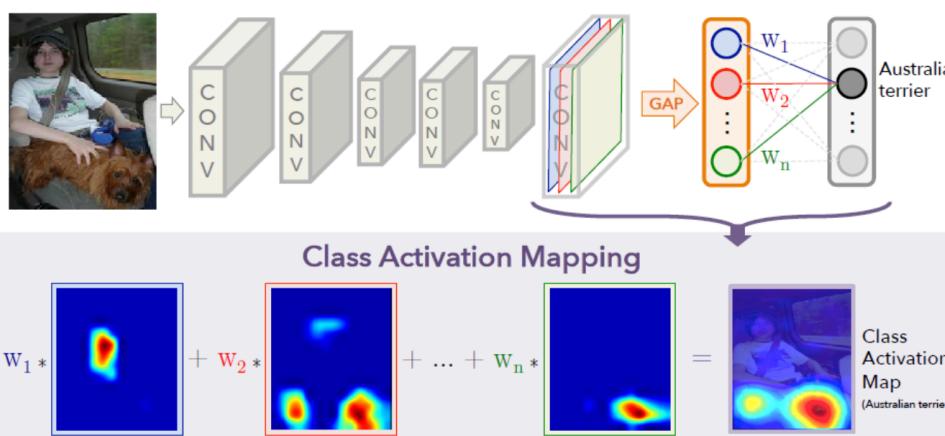
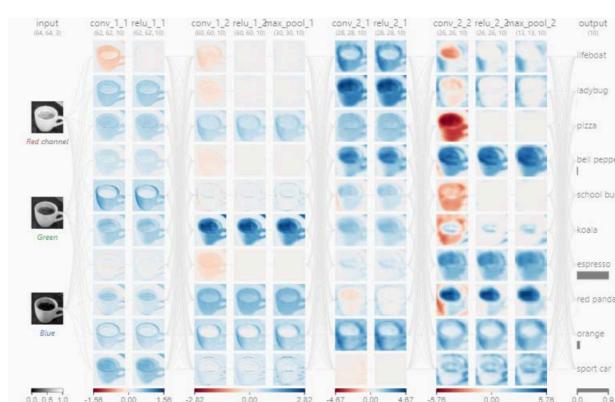
Дополнительным важным направлением является оценка чувствительности влияния значений входных параметров на выход сети. Для этого значения выбранного входа варьируются в области его определения, в то время как остальные параметры остаются фиксированными и отслеживаются изменения в выходе сети. Знания, полученные из этой формы анализа, могут быть представлены в виде таких правил, как:

«ЕСЛИ X уменьшается на 5%, ТО Y увеличивается на 8%».

60. Визуализация данных и результатов работы построенной модели.

Обобщение и интерпретация функционирования сверточных слоев

Одним из подходов, для обобщения и интерпретации результатов функционирования сверточных слоев является метод Class Activation Mapping (CAM), который представляет собой взвешенную карту активации, созданную для каждого изображения, что помогает определить область, на которую акцентирует внимание НС, при классификации изображения.



Методы интерпретации

GradCAM и Vision Transformer

Интерпретируемость двух популярных архитектур глубокого обучения — ResNet50 и Vision Transformer (ViT-224) — при задаче классификации микроскопических изображений бактерий. Представлена работа встроенных карт-внимания Vision Transformer и пост-интерпретация с помощью Grad-CAM для ResNet50. Показаны тепловые карты с выделенными зонами с наибольшим влиянием на прогноз модели.

Механизм внимания демонстрирует более сконцентрированные участки, однако в обоих способах интерпретации работы моделей наблюдаются фоновые зоны с высокой значимостью на прогноз модели. Это свидетельствует, что фоновый шум может иметь влияние на формирование прогноза, что в целом негативно влияет на способность генерализации моделей.

