1. Что такое машинное обучение? Кто такой Data Scientist? Как машинное обучение и наука о данных связаны с искусственным интеллектом?

МО - наука (и искусство) программирования вычислительных машин таким образом, чтобы они могли учиться на основе данных. Более общее определение: МО - область знаний, которая изучает способы обучения вычислительных машин без явного программирования. Определение для инженера: Говорят, что компьютерная программа обучается на опыте E в отношении некоторой задачи T и некоторой меры производительности P, если её производительность в задаче T, измеряемая с помощью P, улучшается с накоплением опыта T. Машинное обучение - это - математика + статистика + программирование - алгоритмический подход к обработке (больших) данных - слабый искусственный интеллект Машинное обучение не является: - полноценным искусственным интеллектом - осведомленным о предметной области - панацеей от всех проблем человечества

Искусственный интеллект - это интеллект, демонстрируемый машинами, в отличии от естественного интеллекта, проявляемого животными, включая людей. …

data scientist - hacking skills ∩ math & statistics knowledge ∩ substantive expertise machine learning - hacking skills ∩ math & statics knowledge traditional research - math & statistics knowledge ∩ substantive expertise danger zone - hacking skills ∩ substantive expertise # 2. Уровни развития искусственного интеллекта (слабый, сильный, ANI, AGI, ASI). ANI - artifical narrow intelligence aka weak ai - narrow capability - present (искусственный узкий интеллект, он же слабый ИИ - узкие возможности - присутствуют) AGI - artifical intelligence aka strong ai - general capability - future? (искусственный интеллект, он же сильный ИИ - общие возможности - будущее?) ASI - artifical super intelligence aka strong ai - transcedent capability - possible? (искусственный сверхинтеллект, он же сильный ии - запредельные возможности - возможны?)

3. История развития ИИ, МО и глубокого обучения

1950 - 1980: Искусственный интеллект (ранний искусственный интеллект вызывает волнения). инжиниринг создания интеллектуальных машин и программ 1951 - 2010: Машинное обучение (МО начинает процветать). Способность к обучению без явного программирования. 2011 - н.в.: Глубокое обучение (Прорыв в глубоком обучении привел к буму искусственного интеллекта). Обучение на основе глубокой нейронной сети

4. В каких областях применяется машинное обучение? Приведите примеры решения прикладных задач с помощью МО.

Распознавание изображений (турникет ГУАП), автоматический переводчик, медицинская диагностика, торговля на фондовом рынке, предотвращение онлайн мошенничества, виртуальный ассистент, фильтр спама, машины с автопилотом, продуктовая рекомендация, предсказание трафика, распознавание речи и др.

5. Постановка задачи обучения на примерах.

X - множество объектов Y - множество ответов (предсказаний, оценок, прогнозов) $\phi(x), \phi : X \to Y$ - неизвестная зависимость (целевая функция) Дано: $\{x_1, \ldots, x_l\} \subset X$ - обучающая выборка $y_i = \phi(x), i = 1, \ldots, l$ - известные ответы Найти: - $g(x, \theta), g : X \times \Theta \to Y$ - алгоритм, функция принятия решений или параметрическая модель, приближающая ϕ на всей выборке X. - $\theta \in \Theta$ - вектор параметров модели, такой, что $g(x, \theta) \approx \phi(x)$

6. Описание объектов и ответов. Типы задач машинного обучения.

Объекты: $f_j : X \to D_j, j = 1, \ldots, n$ - признаки объектов Типы скалярных объектов: - D_j = { 0, 1 } - бинарный признак f_j; - $|D_j| < \infty$ - номинальный признак f_j; - $|D_j| < \infty$, D_j - упорядоченно-порядковый признак f_j; - $D_j = R$ -

количественный признак f_j: интервал или число. Вектор ($f_1(x)$, ... , $f_n(x)$) - признаковое описание объекта x. Матриц признаков: F = $||f_j(x_i)||_{l\times n}$ = ($f_1(x_1)$ ... $f_n(x_1)$. (... . ($f_1(x_l)$ ... $f_n(x_l)$ **Ответы:** Задачи обучения с учителем: Заданы “ответы учителя” $y_i = \phi(x_i)$ на обучающих x_i Для классификации: - Y = { -1, +1 } - бинарная классификация (2 класса); - Y = { 1, ..., M } - классификация между M не пересекающимися классами; - Y = { 0, 1 }^M - M классов, которые могут пересекаться Для регрессии: - Y = R or Y = R^M.

Ранжинирование: - Y - конечное отсортированное множество

Задачи обучения без учителя - ответов нет, но требуется что-то сделать с самими объектами

Типы задач МО: Статистическое обучение с учителем: - обучение по прецедентам; - восстановление зависимости по эмпирическим данным; - прдсказательное моделирование; - аппроксимация функций по заданным точкам Два основных типа задач - классификация и регрессия

7. Обучение с учителем, предсказательные модели. Приведите не менее 4-х примеров описания прикладных задач.

Обучение с учителем предполагает наличие размеченных данных, где каждой входной характеристике соответствует целевая метка. Вот примеры прикладных задач для предсказательных моделей Модель - параметрическое семейство функций A = $\{g(x,\theta)|\theta \in \Theta\}$, где g: $x \times \Theta \to Y$ - фиксированная функция, Θ - множество допустимых значений параметров θ Пример: Линейная модель с векторным параметров $\theta = (\theta_1, \ ... , \ \theta_n) \in R^n$: $g(x,\theta) = \sum_{j=1}^{n} \theta_j f_j(x)$ - для регрессии и ранжирования, Y = R; $g(x,\theta) = sign \sum_{j=1}^{n} \theta_j f_j(x)$ - для классификации, Y = { -1; +1 } **Примеры:** 1) Кредитный скоринг (классификация) - оценка вероятности того, что клиент не вернет кредит. Данные: возраст, доход, кредитная история, уровень задолженности. Целевая переменная: классы “надежный” или “ненадежный”. 2) Прогнозирование спроса на продукцию (регрессия) - оценка объема продаж товара в следующем месяце. Данные: исторические продажи, сезонность, акции, тренды. Целевая переменная: числовое значение объема продаж. 3) Диагностика заболеваний (классификация) - определение наличия заболевания по медицинским данным.

Данные: результаты анализов, симптомы, история болезни. Целевая переменная: наличие или отсутствие заболевания. 4) Распознавание рукописного текста (классификация) - преобразование изображений рукописных символов в текст. Данные: изображение букв или цифр. Целевая переменная: классы символов или цифр.

8. Алгоритм обучения. Сведение задачи обучения к задаче оптимизации.

Процесс обучения с учителем состоит из 2-х этапов: - Обучение: Алгоритм обучения $\mu : (X \times X)^l \in \Theta$ по выборке $X^l = (x_j, y_j)_{i=1}^l$ строит функцию $g(x, \theta)$, оценивая (оптимизируя) параметры модели $\theta \in \Theta$. $[(f_1(x_1) \ ... \ f_n(x_1))$ $(y_1)]$ (θ_1) ... $\to^{\phi}$ ... $\to^{\mu}$ ... = 0 $[(f_1(x_l) ... f_n(x_l))$ $(y_l)]$ (θ_n) - Применение: Функция $g(x, \theta)$ для новых объектов x'_j выдает ответы $g(x'_j, \theta)$

$(f_1(x'_1) \ ... \ f_n(x'_1))$ $g((x'_1, \theta))$ ... $\to^{g}$ ... $(f_1(x'_k) ... f_n(x'_k))$ $(g(x'_k, \theta))$

Замена задачи обучения на минимизацию: Метод минимизации эмпирического риска: $\mu(X^l) = arg \ min_{g \in A} Q(g, X^l)$ Пример: метод наименьших квадратов, квадратичная ошибка.

9. Оценивание моделей. Эмпирический риск и функция потерь.

Функция потерь □(g, x): для заданного объекта $x \in X$ вычисляет величину ошибки алгоритма (функции) g $\in$ A на этом объекте. Ошибка тем больше, чем сильнее $g(x, \theta)$ отклоняется от правильного ответа $\phi(x)$. Функция потерь для задач классификации: - □(g, x) = $[g(x, \theta) \neq \phi(x)]$ - индикатор ошибки. Функция потерь для задач регрессии: - □(g, x) = $|g(x, \theta) - \phi(x)|$ - абсолютное значение ошибки - □(g, x) = $(g(x, \theta) - \phi(x))^2$ - квадратичная ошибка **Эмпирический риск:** - Нельзя заранее достоверно узнать, на сколько хорошо алгоритм g покажет себя на практике (“риск”), поскольку неизвестной истинный закон распределения данных P(x, y). - Оценить и улучшить работу алгоритма g можно на заранее известной ограниченной обучающей выборке (закон больших чисел). - **Эмпирический риск** - способ оценки качества работы алгоритма g на всей обучающей выборке X^l. - Функционал эмпирического риска: $\frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(g, x_i)$

10. Что такое переобучение (overfitting) и недообучение (underfitting)? Как их можно избежать?

Переобучение - данных мало, параметров слишком много, модель сложная, избыточно гибкая. Ключевая проблема в МО. Из-за чего? - избыточные параметры в модели $g(x, \theta)$ “расходуются” на чрезмерно тонкую подгонку под обучающую выборку; - выбор g из A производится по неполной информации X^l Как обнаружить? - Эмпирически, путем разбиения выборки train и test (для test должны быть известны правильные признаки) Избавиться нельзя, можно минимизировать: - увеличить объем обучающих данных - накладывать ограничения на θ (регуляризация) - минимизировать одну из теоретических оценок - выбирать модель по оценкам обобщающей способности Регуляризация - уменьшение значений параметров (сокращение весов) **Недообучение** - данных много, параметров недостаточно, модель простая, негибкая Из-за чего? - слишком простая модель - недостаточное время обучения - недостаточно информативные признаки Методы борьбы: - усложнение модели - долгое обучение - инженерия признаков - удаление регуляризации или ее ослабление # 11. Одномерная и многомерная линейная регрессия. Л**инейная регрессия** - это метод МО, используемый для прогнозирования числовой целевой переменной на основе одной или нескольких независимых переменных. **Одномерная** - используется, когда есть одна независимая переменная x Мат. модель: $y = \omega_0 + \omega_1 x$, где y - предсказанное значение; x - независимая переменная; ω_1 - коэффициент наклона (угловой коэффициент) ω_0 - свободный член (сдвиг) **Многомерная** - есть несколько независимых переменных $x_1, \ \dots, \ x_n$ Мат. модель: $y = \omega_0 + \omega_1 x + \ \dots, \ + \omega_n x$, где y - предсказанное значение; $x_1, \ \dots, \ x_n$ - независимые переменные $\omega_1, \ \dots, \ \omega_n$ - веса (коэффициенты) ω_0 - свободный член

12. Конструирование признаков.

Использование интуиции для создания новых признаков путем преобразования или комбинирования оригинальных признаков. Пример: предсказание стоимости жилья. Признаки: x_i - площадь квартиры (м. кв.), x_2 - город (категориальный) Модель: $g_1(x, \theta) = \theta_2 x_2 + \theta_1 x_1 + \theta_0$ Добавляем новый признак $x_3 = x_2 x_1$, чтобы напрямую учесть в модели различия стоимости кв. метра в разных регионах. $g_2(x, \theta) = \theta_3 x_3 + \theta_2 x_2 + \theta_1 x_1 + \theta_0$

Способы конструирования признаков: - Полиномиальные признаки: возведение существующих признаков в степень или их комбинирование - Агрегация данных: среднее, сумма или медиана по имеющимся признакам - Лаги - значение за предыдущие периоды, которые могут влиять на текущие - Временные признаки - день недели, месяц или номер квартала, которые учитывают сезонные (периодические) изменения в данных - Знания предметной области.

13. Нормализация признаков

$\tilde{y} = \theta_1 x_1 + \theta_2 x_2 + \theta_0$ - Нормирование значений признаков (нормализация средним): - вычесть математическое ожидание μ_{x_j} признака x_j; - поделить на СКО σ_{x_j} признака x_j;

$$x_i^\star := \frac{x_j - \mu_{x_j}}{\sigma_{x_j}}$$

- Минимаксная нормализация: - значение признака приводятся к диапазону [0, 1] или [-1, 1], например:

$$x_j^\star := \frac{x_j - min\ x_j}{max\ x_j - min\ x_j}$$

- вместо $min\ x_j$ в числителе можно использовать среднее μ_{x_j} или медиану Когда она нужна: - Необходимо стремиться: $-1 \leq x_j \leq 1$ для каждого признака x_j - Эвристика: нормализация не обязательна, если диапазон признака отличается от $-1 \leq x_j \leq 1$ менее, чем на 2 порядка - Нормализация не нужна: $-3 \leq x_1 \leq 3$ $-0.3 \leq x_2 \leq 0.3$ $0 \leq x_3 \leq 3$ $-2 \leq x_4 \leq 0.5$ - Нормализация обязательна: $-100 \leq x_5 \leq 100$ $0.001 \leq x_6 \leq 0.001$ $98.6 \leq x_7 \leq 105$ - Лучше провести нормализацию, чем от нее отказаться

14. Метод наименьших квадратов.

МНК - базовый статистический метод, используемый в линейной регрессии для нахождения оптимальных параметров модели, которые минимизируют ошибку между предсказанными данными и реальными значениями. Идея метода: МНК минимизирует сумму квадратов отклонений предсказанных значений от фактических:

$$error = \sum_{i=1}^{n} (y_i - \widetilde{y_i})^2$$

где y_i - истинное значение, $\widetilde{y_i}$ - предсказанное моделью значение, n - количество наблюдений Достоинства: - простота реализации; - быстро работает на небольших данных; - хорошо интерпретируется; Недостатки: - чувствителен к выбросам; - плохо работает при мультиколлинеарности; - не подходит для сложных нелинейных зависимостей; Как избежать недостатков: 1) Регуляризация снижает влияние шумов и переобучение 2) Удаление выбросов и нормализация данных 3) Полиномиальная регрессия для нелинейных зависимостей

15. Алгоритм градиентного спуска.

Дано: функционал качества $Q(\theta)$ Найти: вектор параметров θ, при котором $Q(\theta) \rightarrow min$ Пример: $min_{\theta_0,\theta_1} Q(\theta_0,\theta_1), \theta(\theta_0,\theta_1)$ **Алгоритм:** 1) Задаем начальное значение для вектора θ (инициализируем веса) Пример $\theta_0 = \theta_1 = 0$ 2) Пошагово изменяем значения элементарного вектора θ, чтобы уменьшить $Q(\theta)$, до тех пор, пока не достигнем (окажемся вблизи) минимума Пример:

$$\theta_0 = \theta_0 - \alpha \frac{g}{g\theta_0} Q(\theta_0,\theta_1)$$

$$\theta_1 = \theta_1 - \alpha \frac{g}{g\theta_1} Q(\theta_0,\theta_1)$$

ВАЖНО: обновлять θ_j необходимо одновременно для всех j α - скорость обучения - Если α слишком мало, требуется большое количество итераций для сходимости - Если α слишком велико, значение функции потерь может не уменьшаться на каждой итерации и алгоритм может не сойтись к устойчивому минимуму

16. Стохастический градиентный спуск.

Проблема: если обучающая выборка X^l велика $l >> 0$, то каждый шаг градиентного спуска будет требовать большого количества вычислений, а сам алгоритм будет работать медленно **Решение:** Стохастический градиентный спуск - берем по одной паре “объект-ответ” (x_j, y_j) и сразу обновляем вектор θ - градиент вычисляем по функции ошибки □, а не по функционалу качества Q - функционал качества оцениваем по приближенной формуле **Алгоритм:** 1) выбрать объект x_i из X^l случайным образом; 2) вычислить потерю: $\epsilon = \mathcal{L}_i(g, x)$;

например $\epsilon_i = (g(x_i, \theta) - y_i)^2$ 3) сделать градиентный шаг: $\theta_j = \theta_j - \alpha \frac{g}{g\theta_j} \mathcal{L}_i(g, x)$ 4) оценить функционал: $\overline{Q} = \lambda \epsilon_i + (1 - \lambda)\overline{Q}$, где λ - скорость забывания 5) повторять шаги 1-4, пока значения $\overline{Q}$ и/или параметры θ не сойдутся

Достоинства: - легко реализуется - легко обобщается на любые $g(x, \theta)$, □(a, y) - легко добавить регуляризацию - возможно динамическое обучение - на сверхбольших выборках можно получить неплохое решение, даже не обработав все $g(x_i, y_i)$ - подходит для задач с большими данными Недостатки: 1) подбор комплекс эвристик является искусством (не забывать про переобучение, завстревание, расходимость)

17. Варианты инициализации весов и выбора скорости обучения в алгоритме градиентного спуска.

1) θ_j = 0 для всех j = 0, ..., n
2) Небольшие случайные значения: $\theta_j = random\ (-\frac{1}{2n}, \frac{(}{1})2n)$
3) $\theta_j = \frac{<y, f_j>}{<f_j, f_j>}, f_j = (f_j(x_j))_{i=1}^l$ - вектор значений признака Эта оценка θ оптимальна, если: функция потерь квадратична и признаки некоррелированы, $< f_j, f_k >= 0, j \neq k$
4) Обучение по небольшой случайной подвыборке объектов
5) Мультистарт: многократные запуски из разных случайных начальный приближений и выборе лучшего решения **Скорость обучения**
6) Постоянные значения $\alpha = const$
7) Убывающее значение. Сходимость гарантируется (для выпуклых функций) при $\alpha_t \to 0 \sum_{t=1}^{\infty} \alpha_t = \infty$; $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$, в частности можно положить $\alpha_t = \frac{1}{t}$, где t - номер шага.
8) Метод наискорейшего градиентного спуска:

$$\mathcal{L}_i(\theta, \alpha \bigtriangledown \mathcal{L}_i(\theta)) \to\ min_\alpha$$

позволяет найти адаптивную скорость $\alpha^\star$ При квадратичной функции потерь $\alpha^\star = ||x_i||^{-2}$
9) Пробные случайные шаги для “выбивания” итерационного процесса из локальных минимумов.
10) Метод Левенберга-Марквардта (второго порядка) # 18. Использование регуляризации для борьбы с переобучением в алгоритме градиентного спуска. Регуляризация - сокращение значений параметров

- Штраф за увеличение нормы вектора весов:

$$\widetilde{\mathcal{L}_i}(\theta) = \mathcal{L}_i(\theta) + \frac{\tau}{2}||\theta||^2 = \mathcal{L}_i(\theta) + \frac{\tau}{2}\sum_{j=1}^{n}\theta_{j=1}^2 \rightarrow \ min_\theta$$

- Градиент: $\bigtriangledown\widetilde{\mathcal{L}_i}(\theta) = \bigtriangledown\mathcal{L}_i(\theta) + \tau\theta$.
- Модификация градиентного шага: $\theta = \theta(1 - a\tau) - a\bigtriangledown\mathcal{L}_i(\theta)$.
- Методы подбора коэффициента регуляризации τ:

1) скользящий контроль;
2) стохастическая адаптация;
3) двухуровневый байесовский вывод.

19. Повышение производительности с помощью векторизации.

Векторизации - техника оптимизации, которая заменяет циклы и поэлементные операции на матричные и векторные вычисления. Она значительно ускоряет обработку данных в задачах МО. Эффективность: - параллельные вычисления: операции над векторами и матрциами выполняются быстрее за счет оптимизации на уровне процессора - устраняются циклы, что снижает затраты на интерпретацию кода - использование оптимизированных библиотек: numpy, tensorflow, pytorch использует высокоэффективные реализации операций

$$g(x_i, \theta) = \theta_0 + \sum_{j=1}^{n}\theta_j x_{i,j} = \sum_{j=0}^{n}\theta_j x_{i,j}$$

где x_i - вектор признаков i-го объекта, θ - вектор параметров $x_{i,0}$ = 1 - фиктивный признак

```
for j in range(0, n):
    g = g + theta[j] * x[j]
```

В векторном виде: $g(x_i, \theta) = \theta \cdot x_i$ - быстрее на порядки

```
import numpy as np
g = np.dot(theta, x)
```

20. Постановка задачи бинарной и многоклассовой классификации.

Классификация - это задача прогнозирования категориальной (дискретной) целевой переменной на основе входных данных В

зависимости от количества классов классификация делится на: - бинарную - многоклассовую **Бинарная** Цель: разделить объекты на 2 класса (0 или 1, да или нет) Формальная постановка: дана обучающая выборка $\{(x_1, y_1), (x_2, y_2), \ ..., \ (x_n, y_n)\}$ где $x_i \in R$ - вектор признаков объекта $y_i \in \{0, 1\}$ - целевая метка класса Цель: найти функцию $f(x)$ такую, что: $f(x) \approx y$, $f(x) = \{0, 1\}$ Примеры: - кредитный скроллинг - спам-фильтр - диагностика заболеваний

Многоклассовая Цель: разделить объекты на 3 или более группы Формальная постановка: Такая же обучающая выборка. $y_i \in \{1, 2, ..., K\}$ - целевая метка из K классов Цель: найти такую функцию, что предсказанное значение близко к действительному. Примеры: - распознавание рукописных цифр - классификация изображений - анализ тональности текста

21. Классификаторы на основе разделяющей поверхности (margin-based).

Классификаторы на основе разделяющей поверхности (margin-based classifiers) — это модели, которые ищут оптимальную гиперплоскость для разделения классов с максимальным отступом (**margin**) между классами. Эти алгоритмы стремятся не просто разделить классы, но сделать это с максимальной уверенностью. **Ключевая идея:** Максимизировать расстояние (**margin**) между разделяющей гиперплоскостью и ближайшими точками обучающей выборки.

Классификаторы на основе разделяющей поверхности фокусируются на максимизации отступа между классами, что делает их устойчивыми к переобучению и обеспечивает высокую точность на сложных данных. Наиболее яркий представитель — **SVM**, эффективно работающий с линейными и нелинейными задачами.

22. Логистическая регрессия

Логистическая регрессия — это алгоритм машинного обучения для решения задач **классификации**, который прогнозирует вероятность принадлежности объекта к определённому классу. Несмотря на название, это не регрессионный, а **классификационный** метод.

В отличие от линейной регрессии, логистическая регрессия предсказывает вероятность, что объект принадлежит к классу 1.

Для этого используется **сигмоидная функция** (логистическая функция), которая преобразует любое значение в диапазон (0,1)(0, 1)(0,1).

23. Принцип максимизации правдоподобия, его связь с эмпирическим риском

Максимизация правдоподобия заключается в выборе параметров модели, которые делают наблюдаемые данные максимально вероятными.

Для набора данных {x1,x2,...,xn} и модели с параметрами θ **Идея:**
Эмпирический риск — это среднее значение потерь на обучающей выборке. Модель обучается так, чтобы минимизировать эту ошибку.

Максимизация правдоподобия и **минимизация эмпирического риска** тесно связаны, особенно в вероятностных моделях.

Для вероятностной модели можно взять функцию потерь как **отрицательный логарифм правдоподобия Примеры связи**
1. **Логистическая регрессия:**

- ** ** .
- - **Log Loss**, .

2. **Линейная регрессия:**

   - При предположении, что ошибки имеют **нормальное распределение**, максимизация правдоподобия приводит к **методу наименьших квадратов**. **#** 24. L1- и L2-регуляризация. Вероятностный смысл регуляризации. Двухуровневая модель порождения данных: $P(y|x;\theta)$ - вероятностная модель данных; $p(\theta,\gamma)$ - априорное распределение параметров модели γ - вектор гиперпараметров В этой модели случайной (стохастической) является не только выборка X^l, но и вектор параметров θ, а значит и $g(x,\theta)$ Совместное правдоподобие данных и модели: $p(X^l,\theta) = p(X^l|\theta)p(\theta,\gamma)$ Принцип максимума апостериорной вероятности (Maximum a Posteori Probability, MAP):

$$Q_{MAP}(\theta) = ln\, p(X^l,\theta) = \sum_{i_1}^{l} log\, P(y_i|x_i;\theta) + log\, p(\theta;\gamma) \to max_\theta$$

Регуляризация — это метод, который предотвращает **переобучение** модели, добавляя штраф за сложность модели к функции потерь. Она помогает контролировать величину весов модели, что делает её более устойчивой и обобщаемой. L_1 Добавляет к функции потерь сумму **модулей весов Особенности:**

- **Спарсити (разреженность):** обнуляет малозначимые веса, выполняя **автоматический отбор признаков**.
- Хорошо работает, когда есть много неинформативных признаков. L_2 Добавляет к функции потерь сумму **квадратов весов Особенности:**
- **Сглаживает веса:** уменьшает влияние всех признаков, но редко обнуляет веса полностью.
- Предпочтительна при **мультиколлинеарности** (когда признаки скоррелированы). Пример L_1 и L_2 регуляризации
- Пусть параметры θ_j независимы, E_{θ_j} = 0, D_{θ_j} = C
- Распределение Гаусса и квадратичный L_2 регуляризатор
- Распределение Лапласа и абсолютный (L_1) регуляризатор
- C - гиперпараметр, $\tau = \frac{1}{C}$ - коэффициент регуляризации

25. Понятие расстояния между объектами. Метрика Минковского.

В задачах машинного обучения и анализа данных важно измерять **сходство** или **различие** между объектами. Это достигается с помощью **метрик расстояния**, которые определяют, насколько два объекта “близки” друг к другу в пространстве признаков. - **Неотрицательность:** $d(x, y) \geq 0$ (расстояние не может быть отрицательным) - **Тождественность:** $d(x, y) = 0 \iff x = y$ (расстояние равно нулю только между одинаковыми объектами) - **Симметричность:** $d(x, y) = d(y, x)$ (расстояние одинаково в обе стороны) - **Неравенство треугольника:** $d(x, z) \leq d(x, y) + d(y, z)$ (кратчайший путь между двумя точками не длиннее любого обходного пути)

Метрика Минковского обобщает несколько популярных мер расстояния и задаётся формулой:

$$d_p(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

где - x=(x1,x2,...,xn) и y=(y1,y2,...,yn) — объекты в nnn-мерном пространстве, - p≥1 — параметр, определяющий тип метрики.

Частные случаи метрики Минковского:

1. **Манхэттенское расстояние (L1-норма, p=1p = 1p=1)**: $d_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|$ (расстояние вдоль осей, как по кварталам города)
2. **Евклидово расстояние (L2-норма, p=2p = 2p=2)**: $d_2 2(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ (прямое линейное расстояние)
3. **Чебышёвское расстояние (**$p \to \infty p \to \infty$**)**: $d_\infty(x,y) = \max_i |x_i - y_i|$ (максимальное различие по одной из координат)

- **Манхэттенская метрика (L1):**
  Лучше работает с разреженными данными или данными с выбросами.
- **Евклидова метрика (L2):**
  Эффективна для данных без выбросов и с равномерно масштабированными признаками.
- **Чебышёвская метрика:**
  Применяется, когда важен **максимальный** признак (например, в шахматах для хода ладьи).

26. Обобщенный метрический классификатор. Метод k ближайших соседей.

Обобщённый метрический классификатор — это класс методов машинного обучения, которые используют понятие расстояния для классификации объектов. Классификация происходит на основе **сходства** между объектами: чем ближе объект к другому объекту в пространстве признаков, тем более вероятно, что они принадлежат к одному классу. **Общая идея:** Для объекта xxx вычисляется расстояние d(x,xi) до объектов обучающей выборки. На основе этих расстояний принимается решение о принадлежности объекта к определённому классу. - Для произвольного $x \in X$ отранжируем объекты $x_1, \ldots, x_l$: $rho(x, x_{(1)}) \leq \rho(x, x^{(2)}) \leq \ldots \leq \rho(x, x^{(l)})$, где x^i i-ый сосед объекта x среди $x_1, \ldots, x_l$; $y^{(i)}$ - ответ на i-м соседе объекта x. - Метрический алгоритм классификации относит объект x к тому классу, которому принадлежит его ближайшие соседи:

$$g(x; X^l) = arg\max_{y \in Y} \sum_{i=1}^{l} [y^{(i)} = y | w(i,x)]$$

. $_y(x)$ $w(i,x)$ - вес, степень близости к объекту x его i-го соседа, неотрицателен, не возрастает по i. $_y(x)$ - оценка близости объекта x к классу y.

Метод k ближайших соседей (kNN): Метод **k-ближайших соседей** классифицирует новый объект по классам его ближайших соседей в обучающей выборке.

Пошаговый алгоритм: 1. **Выбор метрики расстояния** (например, Евклидово, Манхэттенское). 2. **Поиск kkk ближайших соседей** нового объекта. 3. **Голосование:** объект получает класс, который наиболее часто встречается среди kkk соседей. $w(i,x) = [i \leq 1]$ - метод ближайшего соседа $w(i,x) = [i \leq k]$ - метод k ближайших соседей Преимущества: - простота реализация - параметр k можно оптимизировать по leave-one-hot:

$$LOO(k, X^l) = \sum_{i=1}^{l} [g(x_i; \frac{X^l}{x^i}, k) \neq y_i] \to \min_k$$

Недостатки: - неоднозначность классификации при $_y(x) = _s(x), y \neq s$ - не учитываются значения расстояний

27. Метод k взвешенных ближайших соседей. Метод окна Парзена.

Метод **kkk-взвешенных ближайших соседей** является усовершенствованной версией классического алгоритма **kkk-ближайших соседей (k-NN)**. В отличие от обычного k-NN, который учитывает только количество объектов каждого класса среди ближайших соседей, взвешенный k-NN принимает во внимание **расстояние** до соседей, назначая **больший вес ближним** и **меньший вес дальним**. $w(i,x) = [i \leq k]\theta_i$ θ_i - вес, зависящий только от номера соседа - возможные эвристики: $\theta_i = \frac{k+1-i}{k}$ - линейное убывание веса; $\theta_i = q^i$ - экспоненциально убывающие веса, q < q < 1 Проблемы: - как более основано задать веса? - возможно, было бы лучше, если бы вес w(i, x) зависел не от порядкового номера соседа i, а от расстояния до него $\rho(x, x(i))$ **Преимущества взвешенного k-NN** 1. **Снижение влияния шумов:** дальние и потенциально ошибочные объекты имеют меньший вес. 2. **Гибкость:** ближние соседи оказывают большее влияние на классификацию. 3. **Улучшение точности:** особенно в случаях, когда классы расположены неравномерно.

Метод окна Парзена Метод окна Парзена — это **непараметрический метод оценки плотности вероятности** распределения данных. Этот метод используется для **классификации** и **регрессии** на основе локальной плотности объектов в пространстве признаков.
$w(i,x) = K(\frac{\rho(x,x^{(i)})}{h})$ - где h - ширина окна (bandwidth; радиус окрестности) K(r) - ядро, не возрастает и положительно на [0, 1]
- Метод парзеновского окна фиксированной ширины:

$$g(x;X^l,h,K) = arg\max_{y\in Y}\sum_{i=1}^{l}[y_i = y]K(\frac{\rho(x,x^{(i)})}{h})$$

- Метод парзеноского окна переменной ширины:

$$g(x;X^l,k,K) = arg\max_{i\in Y}\sum_{i=1}^{l}[y_i = y]K(\frac{\rho(x,x^{(i)})}{\rho(x,x^{k+1})})$$

- Оптимизация параметров - по критерию LOO: - выбор ширины окна h или числа соседей k - выбор ядра k # 28. Оптимальная разделяющая гиперплоскость, ее геометрическая интерпретация. В задачах **линейной классификации** необходимо найти такую гиперплоскость, которая **максимально разделяет** объекты разных классов. Эта гиперплоскость делит пространство признаков на две части, каждая из которых соответствует одному из классов.

29. Применение условий Каруша-Куна-Такера к задаче построения оптимальной разделяющей гиперплоскости.

В задаче построения **оптимальной разделяющей гиперплоскости** в **методе опорных векторов (SVM)** необходимо максимизировать зазор между классами. Эта задача сводится к задаче **выпуклой оптимизации** с ограничениями, для решения которой применяются **условия Каруша-Куна-Такера (KKT)**.

Для оптимальности решения должны выполняться следующие условия:

1. **Стационарность (нулевой градиент лагранжиана)**
2. Допустимость (прямые ограничения)
3. Допустимость множителей Лагранжа:
4. Условие дополняющей нежесткости:

Условия Каруша-Куна-Такера гарантируют, что решение задачи SVM оптимально. Только **опорные векторы** влияют на положение гиперплоскости, а остальные объекты данных — нет. KKT-условия позволяют перейти к двойственной задаче, которая решается более эффективно, особенно при использовании **ядровых методов**.

30. Понятие опорного вектора и типизация объектов.

В **методе опорных векторов (SVM)** опорными векторами называют объекты обучающей выборки, которые **находятся на границе разделения классов** или **ближе всего к гиперплоскости**. Именно они определяют положение **оптимальной разделяющей гиперплоскости**.

31. Нелинейное обобщение метода опорных векторов с помощью функции ядра. Виды ядер.

Линейный SVM хорошо работает при линейно разделимых данных, но многие реальные задачи имеют сложную, **нелинейную границу** между классами. Для решения таких задач используется **ядровой метод** (kernel trick), который позволяет строить **нелинейные разделяющие гиперплоскости**.

32. Интерпретируемость алгоритмов машинного обучения

Интерпретируемость — это степень, с которой человек может понять, как и почему модель машинного обучения принимает те или иные решения. **Важно:** Интерпретируемая модель помогает объяснить, какие признаки влияют на предсказания, насколько надежны решения модели и почему возникают ошибки. Это критично в сферах с высокими рисками: **медицина**, **финансы**, **юриспруденция**. **Баланс между точностью и интерпретируемостью**

- **Высокая интерпретируемость** — часто у **простых моделей**, но они могут уступать в точности.

- **Высокая точность** — часто у **сложных моделей** (глубокие нейронные сети, ансамбли), но они менее интерпретируемы.

Почему интерпретируемость важна

1. **Доверие к модели** — прозрачность повышает доверие пользователей.
2. **Выявление ошибок** — легче находить ошибки и улучшать модель.
3. **Соответствие законодательству** — в критичных сферах (например, GDPR требует объяснять автоматические решения).
4. **Обнаружение скрытых зависимостей** — помогает избежать использования нежелательных или неэтичных факторов (например, дискриминация).

33. Деревья принятия решений. Определение, алгоритмы построения.

Дерево решений — это модель машинного обучения, которая принимает решения путем последовательного разветвления данных на основе условий (правил), сформированных из признаков. Оно представлено в виде **иерархической структуры**, где каждый узел соответствует проверке условия по одному из признаков, а листья — финальным решениям или предсказаниям.

34. Критерий Джинни, энтропийный критерий.

При построении дерева решений важно правильно выбирать признаки для разбиения данных. Для этого используются **критерии оценки качества разбиения**, которые измеряют, насколько хорошо разделяются объекты разных классов. Два самых популярных критерия:

1. **Критерий Джини** (Gini Impurity)
2. **Энтропийный критерий** (Information Gain, основанный на энтропии)

1 **Идея:** измеряет вероятность ошибочной классификации случайно выбранного объекта, если его метка определяется случайно в соответствии с распределением классов в подмножестве.
2 **Идея:** измеряет степень неопределенности (энтропии) в данных. Чем ниже энтропия, тем “чище” подмножество. #

35. Проблема переобучения деревьев принятия решений. Регулирование глубины дерева (обрезка ветвей). **Переобучение (Overfitting)** — это ситуация, когда дерево решений слишком точно подстраивается под обучающие данные, включая шум и случайные закономерности. В результате модель показывает отличные результаты на тренировочных данных, но плохо обобщает на новых (тестовых) данных. **Причины переобучения в деревьях решений**

1. **Чрезмерная глубина дерева:**
    - Глубокое дерево может “запомнить” все тренировочные примеры.
    - Каждая ветвь доходит до индивидуальных примеров.
2. **Мелкие разбиения:**
    - Дерево продолжает делить узлы, даже если это не приводит к значимому улучшению.
3. **Много признаков с шумом:**
    - Дерево может использовать неинформативные признаки, подстраиваясь под случайные зависимости.

Методы борьбы: Ограничение глубины дерева Минимальный размер узла Минимальный размер листа обрезка ветвей

36. Объясните понятие ошибок первого и второго рода, их связь с машинным обучением.

В статистике и машинном обучении при принятии решений на основе данных возможны две ключевые ошибки:

1. **Ошибка первого рода (Type I Error) — ложноположительная ошибка (False Positive, FP)**
2. **Ошибка второго рода (Type II Error) — ложноотрицательная ошибка (False Negative, FN)**

1) Ошибка возникает, когда **ложно отвергается** истинная нулевая гипотеза. В терминах машинного обучения — это случай, когда модель **неправильно классифицирует Пример:**

- Модель для диагностики болезни диагностировала болезнь у **здорового** пациента.
- Система антифрода заблокировала **легальную** транзакцию.

2) Ошибка возникает, когда **ложно принимается** ложная нулевая гипотеза. В машинном обучении это означает, что модель **не распознала положительный пример**. **Пример:**

- Модель не выявила болезнь у **больного** пациента.
- Система безопасности не заметила **мошенническую** транзакцию.

Компромисс: Снижение одной ошибки часто увеличивает другую.

- **Уменьшаем FP (ошибку I рода):** Ужесточаем критерий принятия положительного решения → растет FN.
- **Уменьшаем FN (ошибку II рода):** Смягчаем критерий → растет FP.

37. Объясните понятия accuracy, полноты (recall), точности (precision) и F1- меры.

38. Кривые ROC и Precision-Recall, площадь под ними.

39. Метрики оценки качества регрессии.

40. Задача кластеризации. Типы кластерных структур, чувствительность к выбору признаков.

41. Задача частичного обучения.

42. Оценка качества решения задачи кластеризации.

43. Метод k-средних.

44. Алгоритм DBSCAN.

45. Иерархическая кластеризация

46. Карты Кохонена.

47. Ансамбль моделей машинного обучения.

48. Методы стохастического ансамблирования.

49. Случайный лес.

50. Отличие между бэггингом и бустингом.

Бэггинг (Bagging, Bootstrap Aggregating)

- **Идея:** Одновременное обучение множества слабых моделей на случайных подвыборках данных.
- **Как работает:**
  1. Создаются случайные подвыборки исходных данных (с возвращением).
  2. На каждой подвыборке обучается модель (например, дерево решений).
  3. Предсказания объединяются (усредняются или берётся большинство).
- **Цель:** Уменьшить **дисперсию** модели, снизить переобучение.
Бустинг (Boosting)
- **Идея:** Последовательное обучение слабых моделей, где каждая следующая исправляет ошибки предыдущих.
- **Как работает:**
  1. Первая модель обучается на исходных данных.
  2. Ошибки этой модели усиливаются в следующих обучениях.
  3. Итоговый результат — взвешенная комбинация всех моделей.
- **Цель:** Уменьшить **смещение** модели, повысить точность.

51. Алгоритм AdaBoost.

- **Идея:** Улучшать слабые модели, усиливая влияние на сложные для классификации объекты.
- **Как работает:**
  1. Всем объектам присваиваются равные веса.
  2. Слабый классификатор обучается.
  3. Ошибки усиливаются: объекты, которые классифицированы неверно, получают больший вес.
  4. Процесс повторяется.
  5. Итоговое решение — взвешенное голосование слабых моделей.

 Формула итогового классификатора:

$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$ # 52. Градиентный бустинг.

- **Идея:** Модели обучаются последовательно, минимизируя ошибку предсказания с помощью градиентного спуска.
- **Как работает:**
  1. Первая модель делает предсказание.

2. Вычисляется ошибка (остаток) между предсказанием и истинным значением.
3. Следующая модель обучается на этой ошибке.
4. Модели комбинируются, постепенно улучшая результат.

 Формула:

$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$

где:

- $F_m(x)$ — итоговая модель,
- $h_m(x)$— новая модель, обученная на ошибке,
- γ_m — шаг обучения.
- **Примеры: XGBoost**, **LightGBM**, **CatBoost**.

53.Алгоритмы CatBoost, XGBoost.

CatBoost

- Разработан **Яндексом**.
- Оптимизирован для **категориальных признаков** (обработка без One-Hot Encoding).
- Использует **упорядоченный бустинг** для снижения переобучения.
- Быстрый и удобный для задач классификации и регрессии.

XGBoost (Extreme Gradient Boosting)

- Улучшенный градиентный бустинг.
- Использует **регуляризацию** для борьбы с переобучением.
- Оптимизирован по скорости и использованию памяти.
- Хорошо работает с большими и сложными данными.

54. Работа с большими данным. Экосистема Apache Hadoop.

Apache Hadoop

- **Открытая экосистема** для обработки больших данных.
- Масштабируется на тысячи серверов.
- Включает:
  - **HDFS** (файловая система),
  - **MapReduce** (модель вычислений),
  - **YARN** (ресурсный менеджер).

55.Файловая система HDFS.

HDFS (Hadoop Distributed File System)

- **Распределённая файловая система**, разбивает файлы на блоки и хранит их на кластере.
- Поддерживает **избыточность** (репликация данных) для отказоустойчивости.
- Состоит из:
  - **NameNode** — управляет метаданными.
  - **DataNode** — хранит данные.

56.Алгоритм Map-Reduce.

MapReduce

- **Модель обработки больших данных** с разделением на две фазы:
  1. **Map** — разбивает задачу на подзадачи и обрабатывает их параллельно.
  2. **Reduce** — объединяет результаты из Map-фазы. # 57.Apache Spark и распределенные наборы данных (RDD). **Apache Spark** — это платформа для обработки больших данных в реальном времени. Она быстрее Hadoop за счёт работы в памяти и поддерживает разнообразные задачи: SQL, стриминг, машинное обучение и графовые вычисления. **RDD** — это основная абстракция данных в Spark. Это **распределённая коллекция объектов**, которая может быть обработана параллельно на кластере. ### **Особенности RDD:**

1. **Устойчивость (Resilient):** Автоматическое восстановление данных при сбоях.
2. **Распределённость (Distributed):** Данные делятся между узлами кластера.
3. **Неизменяемость (Immutable):** После создания RDD не изменяются, можно только создавать новые.
4. **Ленивые вычисления (Lazy Evaluation):** Операции выполняются только при вызове действия (action). ### **Типы операций с RDD:**

- **Трансформации (Transformations):** создают новые RDD (`map`, `filter`, `flatMap`).
- **Действия (Actions):** возвращают результат (`count`, `collect`, `reduce`).

58.Принципы работы рекомендательных систем.

Рекомендательные системы помогают пользователям находить интересные товары, фильмы, музыку и многое другое.

Принципы работы:

1. **Анализ поведения пользователей:** история покупок, просмотров.
2. **Построение профиля пользователя:** предпочтения, интересы.
3. **Поиск похожих пользователей или объектов.**
4. **Формирование рекомендаций.**

Основные типы рекомендательных систем:

1. **Коллаборативная фильтрация (Collaborative Filtering):** рекомендации на основе поведения похожих пользователей.
2. **Контентная фильтрация (Content-Based Filtering):** рекомендации на основе характеристик товаров.
3. **Гибридные системы:** объединяют оба подхода. # 59. Коллаборативная и контентная фильтрация. ### **Коллаборативная фильтрация (CF):**

Идея: Если пользователь A похож на пользователя B, то A может понравиться то, что нравится B.

- **User-Based:** поиск похожих пользователей.
- **Item-Based:** поиск похожих товаров.

Контентная фильтрация:

Идея: Рекомендации на основе анализа характеристик объектов и предпочтений пользователя.

- **Пример:** Если пользователь смотрел боевики, ему рекомендуются фильмы с тегом “боевик”. # 60. Техники коллаборативной фильтрации: memory-based и model-based. ### **1. Memory-Based (на основе памяти):**

Работает напрямую с матрицей пользователь-объект.

- **User-Based CF:** ищет пользователей с похожими оценками.
- **Item-Based CF:** ищет похожие товары.

Методы сходства:

- **Косинусное сходство:** $sim(A, B)\text{sim}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$
- **Корреляция Пирсона:** $sim(A, B) = \text{sim}(A, B) = \frac{\sum(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum(A_i - \bar{A})^2}\sqrt{\sum(B_i - \bar{B})^2}}$ ### **2. Model-Based (на основе моделей):**

Использует машинное обучение для прогнозирования предпочтений.

- **Методы:** SVD, ALS, градиентный бустинг.

 Пример:
ALS (Alternating Least Squares): разлагает матрицу пользователь-объект для предсказания недостающих оценок.

61. Корреляционные модели в коллаборативной фильтрации. Непараметрическая регрессия, функции сходства.

Корреляционные модели

Используют статистические зависимости между пользователями или объектами.

 Корреляция Пирсона — измеряет линейную зависимость.
 Косинусное сходство — измеряет угол между векторами предпочтений.

Непараметрическая регрессия

Используется для сглаживания и прогнозирования без фиксированной формы функции.

 Примеры:

- **K-ближайших соседей (KNN):** прогнозирует рейтинг на основе ближайших пользователей.
- **Функции ядра:** сглаживание данных с помощью ядровых функций. ## **Функции сходства**

Функции сходства — это методы, которые измеряют, насколько два объекта похожи друг на друга.

 Популярные метрики сходства:

1. **Косинусное сходство:** $\text{sim}(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$

2. **Корреляция Пирсона:**
   Учитывает линейную зависимость между переменными.
3. **Манхэттенское расстояние:** $d(A, B) = \sum_{i=1}^{n} |A_i - B_i|$
4. **Евклидово расстояние:** $d(A, B)\sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$ # 62.Понятие латентной модели. **Латентная модель** — это модель, которая использует скрытые (латентные) переменные для описания наблюдаемых данных. Эти скрытые факторы не наблюдаются напрямую, но влияют на поведение системы.

 Примеры:

- В рекомендательных системах латентные факторы могут представлять интересы пользователей и характеристики товаров.
- В обработке текста латентные переменные могут обозначать темы документа.

 Пример математической модели:
Матрица пользователь–товар RRR разлагается на две латентные матрицы:

$R \approx P \times Q^T$

где:

- P — матрица скрытых предпочтений пользователей,
- Q — матрица скрытых характеристик товаров.

63.Матричные разложения. Сингулярное разложение.

Матричные разложения — это представление матрицы в виде произведения нескольких матриц для упрощения вычислений.

Сингулярное разложение (SVD)

Разлагает матрицу RRR на три матрицы:

$R = U\Sigma V^T$

где:

- U — матрица левых сингулярных векторов (пользователи),
- Σ— диагональная матрица сингулярных значений (важность факторов),
- V^T — матрица правых сингулярных векторов (товары).

Использование в рекомендательных системах:
SVD помогает найти скрытые зависимости между пользователями и товарами для улучшения рекомендаций.

64. Измерение качества рекомендаций.

Для оценки качества рекомендательных систем используют различные метрики.

Основные метрики:

1. **MAE (Mean Absolute Error):**

   $MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i$

   Средняя абсолютная ошибка.

2. **RMSE (Root Mean Square Error):**

   $RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$

   Корень из средней квадратичной ошибки.

3. **Precision@k и Recall@k:**
   Оценивают точность и полноту рекомендаций в топ-kkk списках.

4. **MAP (Mean Average Precision):**
   Средняя точность для всех пользователей.

5. **NDCG (Normalized Discounted Cumulative Gain):**
   Учитывает порядок выдачи рекомендаций. # 65.Понятие искусственного нейрона. **Искусственный нейрон** — это математическая модель биологического нейрона.

Формула:

$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$

где:

- $x_i x$ — входы нейрона,
- w_i — веса,
- b — смещение (bias),
- f — функция активации (ReLU, сигмоида),
- y — выход.

Функции активации:

- **Сигмоида:** $sigma(x) = \frac{1}{1+e^{-x}}$

- **ReLU:** $f(x) = \max(0, x)$ # 66.Сеть Хопфилда. **Сеть Хопфилда** — это рекуррентная нейронная сеть для хранения и восстановления образов.

 Особенности:

- Симметричные веса ($w_{ij} = w_{ji}$).
- Динамическая система с энергией.

 Энергетическая функция:

$E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j + \sum_i \theta_i s_i$

где:

- s_i — состояние нейрона,
- %_i% — порог,
- w_{ij} — веса связей.

 Применение: распознавание образов, исправление ошибок.

67.Понятие стохастического нейрона. Машина Больцмана.

Стохастический нейрон

Решает, активироваться или нет, с вероятностью, зависящей от входного сигнала.

 Формула:

$P(s_i = 1) = \frac{1}{1+e^{-z_i}}$

где z_i— взвешенная сумма входов.

Машина Больцмана (RBM)

Стохастическая нейронная сеть с двумя слоями: видимый и скрытый.

 Энергетическая функция:

$E(v, h) - \sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{ij} h_j$

где:

- v — видимые узлы,
- h — скрытые узлы,
- $w_{ij} $— веса.

 Применение: обучение признаков, генерация данных.

68.Алгоритм имитации отжига.

Имитация отжига — это метод глобальной оптимизации, имитирующий процесс охлаждения металлов.

Идея:

- Начинается с высокой “температуры” и случайных решений.
- Постепенно температура понижается, уменьшается вероятность перехода к худшим решениям.

Вероятность перехода:

$P \exp\left(-\frac{\Delta E}{T}\right)$

где:

- ΔE — изменение энергии,
- T — температура.

Применение: оптимизация маршрутов, задач планирования.

69.Ограниченная машина Больцмана. Алгоритм контрастного расхождения (contrastive divergence).

Ограниченная машина Больцмана (RBM) — это стохастическая нейронная сеть, которая используется для извлечения признаков, уменьшения размерности и как строительный блок для глубоких сетей.

Структура RBM:

- Состоит из **двух слоёв**:
  - **Видимый слой (Visible layer, vvv)** — входные данные.
  - **Скрытый слой (Hidden layer, hhh)** — латентные переменные.
- **Нет связей внутри слоёв:**
  - Никаких связей между нейронами одного слоя.
  - Полносвязные между слоями.

Энергетическая функция:

Определяет вероятность конкретной конфигурации:

$E(v,h) - \sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{ij} h_j$

где:

- v_i — видимые узлы,
- h_j — скрытые узлы,
- a_i и b_j — пороговые значения,
- w_{ij} — веса между слоями.

Вероятностная модель:

Вероятность конфигурации видимых узлов vvv:

$P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$

где Z — нормализующая константа (разделочная функция).

Контрастное расхождение (CD) — это приближённый и быстрый алгоритм обучения RBM.

Идея:

- Минимизировать расхождение между распределением данных и модельным распределением.
- Избегает вычисления сложной нормализующей константы Z.

Алгоритм CD:

1. **Прямой проход:**
   - Задаются входные данные vvv.
   - Вычисляются скрытые узлы h по вероятности: $P(h_j = 1|v) = \sigma\left(b_j + \sum_i v_i w_{ij}\right)$
2. **Обратный проход:**
   - Восстановление видимого слоя ′v′: $P(v'_i = 1|h) = \sigma\left(a_i + \sum_j h_j w_{ij}\right)$
   - Повторное обновление скрытых узлов h.
3. **Обновление весов:**

   $\Delta w_{ij} = \eta\left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}\right)$

   где:

   - η — скорость обучения,
   - $\langle \cdot \rangle$ — математическое ожидание.

 Особенность: CD использует только несколько шагов (обычно 1-2), что делает обучение быстрым.

70. Глубокие сети на основе машины Больцмана.

Глубокие вероятностные сети (DBN) — это многослойные нейронные сети, построенные из нескольких RBM.

 Структура DBN:

- **Иерархия RBM:** каждый слой обучается отдельно как RBM.
- Нижний RBM обучается на данных, остальные — на выходах предыдущих слоёв.
- Верхние слои могут быть связаны рекуррентно или обучаться с помощью обратного распространения ошибки.

 Этапы обучения DBN:

1. **Жадное послойное обучение (Greedy Layer-Wise Training):**
   - Первый слой обучается как RBM.
   - Замораживаются веса и используется для обучения следующего слоя.
2. **Тонкая настройка (Fine-Tuning):**
   - Используется обратное распространение ошибки для общей настройки сети.

 Математическая модель:
Для DBN вероятность входных данных vvv представляется как:

$$P(v) = \sum_{h_1, h_2, \ldots, h_L} P(v|h_1)P(h_1|h_2) \ldots P(h_{L-1}|h_L)P(h_L)$$

где h_i — скрытые слои.

 Преимущества DBN:

- Эффективное обучение благодаря поэтапной настройке.
- Глубокая иерархия признаков.
- Отлично работает для генерации и восстановления данных.