Лекция 1. Введение в машинное обучение

1. Дать определение задачи машинного обучения. Что такое обучающая выборка, модель и функция потерь?

Определение задачи машинного обучения

> Задача машинного обучения — автоматическое **выявление закономерностей** (зависимостей) в данных с помощью специальных алгоритмов (моделей) и их последующего использования для **прогнозирования** или **принятия решений** на новых, ранее не встречавшихся данных.
>
> Формально: по заданной **выборке данных** $(D = \{(x_i, y_i)\}_{i=1}^n)$ (обучающей выборке) найти **функцию** $(f : X \to Y)$, которая наилучшим образом приближает неизвестную целевую зависимость $(y = f^*(x))$, где $(x \in X)$ — объект (признаковое описание), $(y \in Y)$ — целевая переменная (ответ).

Ключевые компоненты

> 1. **Обучающая выборка**:
>   - **Объект (пример)** (x_i) — вектор признаков (features), описывающий i-й объект. Может быть представлен как точка в (R^m) (если (m) признаков).
>   - **Ответ (target)** (y_i) — значение целевой переменной, которое модель должна предсказывать.
> 2. **Модель**:
>   - Математически созданная зависимость признак-значение.
>   - Обычно задаётся **параметрическим семейством функций** $(f(x; \theta))$, где (θ) — вектор параметров модели.
>   - **Цель обучения** — найти такие значения параметров (θ), при которых модель $(f(x; \theta))$ наилучшим образом описывает данные обучающей выборки.
> 3. **Функция потерь** $(L(f(x; \theta), y))$:
>
>   - Функция, которая **измеряет ошибку** (стоимость) предсказания модели на одном объекте. Она оценивает, насколько предсказание $(\hat{y} = f(x; \theta))$ отличается от истинного ответа (y).
>
>   **Эмпирический риск (empirical risk)** — средняя ошибка модели на всей обучающей выборке:

$$Q(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f(\mathrm{x}_i;\theta), y_i)$$

- **Цель обучения в терминах функции потерь** — найти параметры (θ), минимизирующие эмпирический риск (принцип минимизации эмпирического риска):

$$\theta^* = \arg\min_{\theta} Q(\theta)$$

2. Сформулировать математическую постановку задачи обучения с учителем: выборка D = $\{(x_i, y_i)\}_{i=1}^n$, поиск модели $f(x)$, минимизирующей ошибку на данных.

С математической точки зрения обучение с учителем означает, что требуется определить такую функцию $f(x)$, что функция потерь $\mathrm{L}\,(f(x_i), y_i)$ будет стремиться к минимуму. То есть нужно подобрать параметры модели (коэффициенты при признаках) так, чтобы выходные данные минимально отличались от требуемых.
Важно: Цель — не выучить данные наизусть, а обеспечить способность модели к **обобщению** на новые, ранее не встречавшиеся данные (проще говоря, избегать переобучения).

Математическая постановка задачи обучения с учителем

Дана обучающая выборка: $D = \{(x_i, y_i)\}_{i=1}^n$, где:

- $x_i \in \mathrm{X}$ — признаковое описание i-го объекта
- $y_i \in \mathrm{Y}$ — целевая переменная (ответ) для i-го объекта

Требуется найти функцию (модель) $f : \mathrm{X} \to \mathrm{Y}$ из некоторого семейства функций F , которая минимизирует ожидаемую ошибку (риск):

$$R(f) = \mathrm{E}_{(x,y)\sim P}[L(y, f(x))]$$

где $L(y, \hat{y})$ — функция потерь, измеряющая несоответствие между предсказанием $f(x)$ и истинным значением y.

3. Перечислить основные типы задач машинного обучения (обучения с учителем, обучения без учителя, обучения с подкреплением) и привести по одному примеру из каждого типа.

Обучение с учителем

> **Суть:** Алгоритму на вход подается **размеченная выборка** — набор данных, где для каждого объекта известен правильный ответ (целевая переменная). Цель — научиться предсказывать этот ответ для новых объектов.
>
> - **Классификация**
> - **Цель:** предсказать категорию (спам/не спам и т.п.).
> - **Метрики:** Accuracy, Precision/Recall, F1.
> - **Регрессия**
> - **Цель:** предсказать число (например, цену).
> - **Метрики:** MAE, RMSE, R2.

Обучение без учителя

> **Суть:** Алгоритму даются данные без готовых ответов или меток. Цель — найти скрытые структуры, закономерности или сгруппировать данные.
>
> - **Кластеризация**
> - **Пример:** группировка покупателей по поведению на сегменты.
> - **Алгоритмы:** k-means, иерархическая кластеризация, DBSCAN.
> - **Понижение размерности (PCA, SVD)**
> - **Пример:** сжатие данных для визуализации, сохраняя главные особенности.

Обучение с подкреплением

> **Суть:** Алгоритм (агент) обучается, взаимодействуя со средой. Он совершает действия, получает за них вознаграждение (или штраф) и корректирует стратегию, чтобы максимизировать совокупную награду.
>
> -**Цель:** максимизировать суммарную награду.
>
> - **Примеры:** игра в шахматы/го, управление роботом, настройка систем рекомендаций.

4. Объяснить различие между задачами классификации и регрессии. Какие метрики качества используются в этих задачах (Accuracy, Precision, Recall, F1, MAE, RMSE, R2)?

- **Классификация** — правильно определить метку (категорию) класса. Пример: спам/не спам, определение объекта на изображении.
- **Регрессия** — предсказание непрерывного числового значения. Пример: прогноз цены квартиры по ее характеристикам.

Пример для наглядности:

- **Регрессия:** предсказать **цену** дома на основе его характеристик (площадь, район, год постройки).
- **Классификация:** определить **тип** дома (новостройка/старый фонд) на основе его цены и других характеристик.

Метрики качества:

Метрики для классификации

Метрика	Формула	Описание
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Доля верно предсказанных объектов. Хороша на сбалансированных данных.
Precision	$\frac{TP}{TP+FP}$	Доля истинно положительных среди всех объектов, которые модель назвала положительными. "Насколько мы можем доверять, когда модель говорит 'да'?"
Recall	$\frac{TP}{TP+FN}$	Доля реально положительных объектов, которые модель обнаружила. "Какую часть всех реальных 'да' мы нашли?"
F1-Score	$2 \times \frac{P \times R}{P+R}$	Гармоническое среднее Precision и Recall. Балансирует оба показателя.

Метрики для регрессии

Метрика	Формула	Описание
MAE	$\frac{1}{n}\sum_{i=1}^{n}\|y_i - \hat{y}_i\|$	Средняя абсолютная ошибка. Проста для интерпретации.
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	Среднеквадратичная ошибка. Штрафует за большие отклонения сильнее, чем MAE.
R^2	$1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$	Коэффициент детерминации. Показывает, насколько хорошо модель объясняет дисперсию данных по сравнению с простым средним.

Интерпретация R^2:

- $R^2 = 1$ — идеальная модель.
- $R^2 = 0$ — модель работает не лучше, чем предсказание средним значением.
- $R^2 < 0$ — модель работает хуже тривиального предсказания.

5. Объяснить смысл разбиения данных на обучающую, валидационную и тестовую выборки. Какую задачу решает каждая из этих частей?

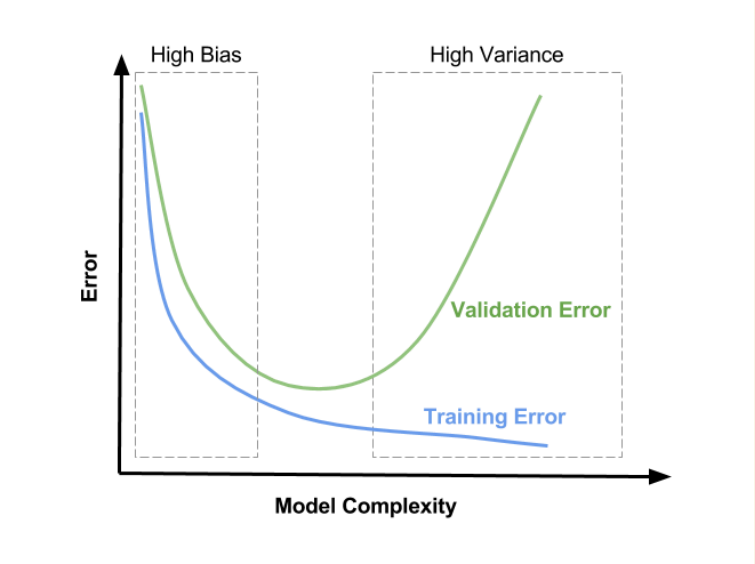
Разбиение данных — фундаментальная практика для оценки **способности модели к обобщению** (работе на новых данных) и борьбы с **переобучением**.

Выборка	Доля (примерно)	Задача
Обучающая (Train)	60-70%	**Обучение модели.** На этих данных алгоритм непосредственно подбирает параметры (веса, коэффициенты).
Валидационная (Validation)	15-20%	**Настройка гиперпараметров и выбор модели.** Оценка обобщающей способности *в процессе* обучения, чтобы вовремя остановиться или выбрать лучшую модель.
Тестовая (Test)	15-20%	**Финальная оценка.** Имитация "боевых" условий. Используется **единожды** в самом конце для объективной оценки итоговой модели на данных, которых она "никогда не видела".

6. Что такое переобучение и недообучение модели? Как они проявляются на графике зависимости ошибки от сложности модели на обучающей и валидационной выборках?

	Переобучение (Overfitting)	Недообучение (Underfitting)
Суть	Модель слишком сложна и "запомнила" обучающие данные, включая шум и выбросы. Не может обобщать.	Модель слишком проста и не уловила основные закономерности в данных.
Причины	Избыток параметров, мало данных, слабая регуляризация, слишком долгое обучение.	Недостаток параметров, слишком сильная регуляризация, мало признаков.
Ошибка на обучении	**Очень низкая** (может стремиться к 0)	**Высокая**
Ошибка на валидации	**Высокая** (значительно выше, чем на обучении)	**Высокая** (близка к ошибке на обучении)

График зависимости ошибки от сложности модели



- **Зона недообучения:** Обе ошибки (обучения и валидации) высоки и близки.
- **Оптимальная сложность:** Ошибка валидации достигает минимума.
- **Зона переобучения:** Ошибка обучения продолжает падать, а ошибка валидации — расти. Кривые расходятся.

7. Дать определение признаков (features) и целевой переменной (label/target). Что такое предобработка признаков и feature engineering? Привести примеры.

- **Признаки (Features)** — это входные переменные, характеристики объектов, на основе которых модель делает предсказания.
- **Целевая переменная (Label/Target)** — это то, что мы хотим предсказать, выход модели, зависимая переменная.

> ### Предобработка признаков (Feature Preprocessing)
>
> Технический этап подготовки "сырых" данных к подаче в модель. Цель — исправить проблемы и привести данные к виду, понятному алгоритмам.
>
> **Примеры:**

1. **Обработка пропусков**: удаление, заполнение средним/медианой (или более сложным алгоритмом)
2. **Кодирование категориальных признаков:**
   - **Label Encoding:** ["низкий", "средний", "высокий"] → [0, 1, 2]
   - **One-Hot Encoding:** Цвет ["красный", "синий"] → `is_red: [1, 0]`, `is_blue: [0, 1]`.
   - **Word Embeddings:** преобразование слов в векторы с сохранением семантики.
3. **Масштабирование:** `StandardScaler` (mean=0, std=1) или `MinMaxScaler` ([0, 1]).
4. **Обработка выбросов:** обрезание, winsorization.
5. **Преобразование распределений:** логарифмирование для скошенных данных.

Feature Engineering

Творческий этап создания **новых признаков** из существующих или внешних данных, чтобы помочь модели лучше выявить закономерности.

Примеры:

- **Из даты:** извлечение дня недели, месяца, является ли выходным.
- **Из текста:** создание признаков "длина текста", "количество определённых слов".
- **Взаимодействие признаков:** умножение "площади" на "цену за кв.м.".

8. Описать алгоритм k-ближайших соседей для задачи регрессии: формула предсказания при равномерных весах и при весах, зависящих от расстояния, роль метрики и гиперпараметра k.

Алгоритм **k-NN для регрессии** предсказывает значение для нового объекта как агрегацию (обычно среднее) значений его `k` ближайших соседей в пространстве признаков.

1. С равномерными весами (Uniform)

Предсказание — простое среднее арифметическое целевых значений `k` соседей.

$\hat{y}(x_{\text{new}}) = \frac{1}{k} \sum_{i=1}^{k} y_i$

Пример

Цены 5 ближайших квартир: `[8, 9, 10, 11, 12]` млн руб.

`ŷ = (8+9+10+11+12) / 5 = 10` млн руб.

2. С весами, зависящими от расстояния (Weighted)

> Близкие соседи вносят больший вклад. Вес w_i обратно пропорционален расстоянию d_i (часто $w_i = \frac{1}{d_i}$).
>
> ***Пример***
> Если ближайшая квартира (8 млн) в 0.1 км, а самая дальняя (12 млн) — в 1 км, то предсказание сместится к 8 млн.

Роль метрики расстояния

Определяет понятие "близости" объектов. Разные метрики дают разных соседей.

- **Евклидова:** $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ — стандартный выбор для непрерывных признаков.
- **Манхэттенская:** $\sum_{i=1}^{n}|x_i - y_i|$ — устойчивее к выбросам.
- **Косинусная:** $1 - \cos(x, y)$ — для текстов, важна направленность векторов.

Важно: Перед использованием k-NN признаки необходимо **масштабировать**, иначе признак с большим размахом (например, зарплата) будет доминировать в расстоянии.

Роль гиперпараметра `k`

- **Малое `k` (k=1):** Модель становится чувствительной к шуму и выбросам → высокий риск **переобучения**.
- **Большое `k`:** Предсказание становится более сглаженным, но может игнорировать локальные особенности → риск **недообучения**.
- **Выбор `k`:** Оптимальное значение подбирается на **валидационной выборке**.

9. Записать правило обновления параметров при градиентном спуске Θ <- Θ - η∇L(Θ). Пояснить смысл шага обучения η и градиента ∇L(Θ)

> *Градиент - это как производная, но для функции по нескольким аргументам (вернее, производная - градиент для функций с одним аргументом)*
>
> Символом $\nabla f(x_1, x_2, \dots)$ (набла) обозначается вектор производных по каждому аргументу

θ_i - вес для feature x_i
θ - вектор весов для вектора x
$\nabla L(\theta)$ - градиент функции потерь для вектора всех весов
η - гиперпараметр, Learning rate

$\theta - \eta\nabla L(\theta) \to \theta$ означает, что новые и более эффективные веса мы будем находить путём вычитания из уже подобранных весов градиента (производной) в этой точке.

В лучшей точке, где функция потерь достигает локального (а желательно глобального) минимума ($\nabla = 0$), мы найдём такой вектор весов, который будет давать лучшие результаты (так как весов, для которых ошибка меньше - нет, мы уже в самой "низкой" по ошибке точке)

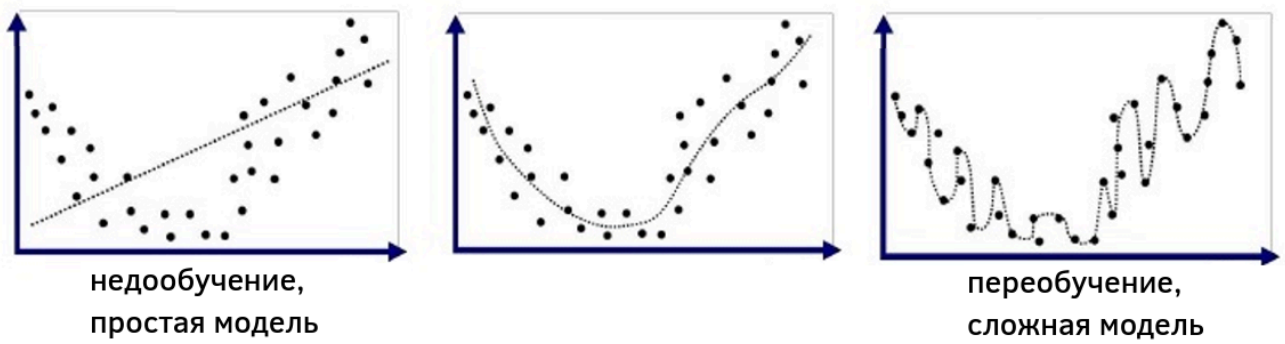
> Смысл градиента ($\nabla L(\theta)$): чем сильнее рост ошибки (производная/градиент) в выбранной точке, тем правильнее будет "спустится" в ту сторону, где ошибка меньше.
> Если производная больше нуля, то ошибка растёт при увеличении x, соответственно "слева" от нынешней точки ошибка будет ниже.
>
> *Выбор градиента обоснован тем, что вблизи минимума функция чаще всего пологая, а потому если производная в точке большая, то мы далеко от минимума - логично вычитать большое значение. Т.е. чем меньше градиент, тем ближе мы к минимуму*

> Смысл learning rate (η): контроль за влиянием градиента на получаемый вес.
> Если η высок, то мы, скорее всего, быстрее найдём минимум (в том числе и локальный), однако рискуем потерять глобальный или более выгодный локальный минимум.
> Если η слишком низок, то градиент оказывает меньшее значение, из-за чего легче "забраться и не вылезти" из какого-то локального минимума

Лекция 2. Линейная регрессия, Ridge и Lasso

Общее про регуляризацию:

Регуляризация необходима для того, чтобы снизить риски переобучения. Когда модель переобучена, некоторые ее веса огромные по модулю, таким образом модель очень "извилистая". А регуляризация применяет штраф на эти веса (или же излишнюю сложность) и модель становится более гладкой.



1. Записать модель линейной регрессии в векторной форме. Объяснить смысл параметров w и b

$$\hat{y} = \mathrm{w}^\top \mathrm{x} + b$$

где:

- w — вектор весов (коэффициентов)
- x — вектор признаков объекта
- b — свободный член (смещение, intercept)
- $\hat{y}$ — предсказанное значение

2. Сформулировать цель обучения линейной регрессии: какие параметры ищутся и по какому критерию они выбираются.

Цель обучения

> Найти такие значения параметров модели (вектора весов (w) и смещения (b)), которые минимизируют ошибку предсказаний модели на обучающих данных.

Искомые параметры

> 1. w - вектор весов (коэффициентов)
> 2. b - свободный член (смещение, bias)

Критерий выбора параметров

> Параметры выбираются путем минимизации функции потерь $L(\hat{y}, y)$, где $\hat{y}$ - предсказанные значения, а y - истинные

Классически используется **MSE** (mean squared error)

$$\mathrm{MSE}(\mathrm{w}, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Процесс обучения

> 1. **Инициализация** параметров (например, нулями или случайными значениями).
> 2. **Вычисление предсказаний** по текущим параметрам: ($\hat{y}_i = \mathrm{w}^\top \mathrm{x}_i + b$).
> 3. **Вычисление значения функции потерь** (MSE) на всей выборке.

4. **Корректировка параметров** с использованием методов оптимизации (например, **градиентный спуск**), чтобы уменьшить MSE.
5. Повторение шагов 2-4 до сходимости (достижения минимума ошибки или выполнения критерия остановки).

3. Записать функционал Ridge-регрессии (L2-регуляризации) и объяснить, как L2-штраф влияет на значения коэффициентов и устойчивость модели.

Регуляризация

Регуляризация необходима для того, чтобы снизить риски переобучения. Когда модель переобучена, некоторые ее веса огромные по модулю, таким образом модель очень "извилистая". А регуляризация применяет штраф на эти веса (или же излишнюю сложность) и модель становится более гладкой.

Функционал Ridge-регрессии

Задача: минимизация J_{ridge}

$$J_{\text{ridge}}(w, b) = \text{MSE}(w, b) + \lambda \|w\|_2^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - (w^\top x_i + b))^2 + \lambda \sum_{j=1}^{d} w_j^2$$

где:

- $\text{MSE}(w, b)$ — среднеквадратичная ошибка
- $\lambda > 0$ — коэффициент регуляризации
- $\|w\|_2^2$ — L2-норма вектора весов (сумма квадратов)

Влияние L2-штрафа

1. **"Сжатие" коэффициентов**: Все веса уменьшаются пропорционально, приближаясь к нулю, но не обнуляются полностью.
2. **Снижение дисперсии**: Уменьшает переобучение, делая модель более устойчивой.
3. **Борьба с мультиколлинеарностью и устойчивость к шуму**: При наличии сильно коррелирующих признаков(мультиколлинеарность), L2 делит вес между ними.
4. **Контроль переобучения**: Чем больше λ, тем сильнее "сжимаются" веса, тем более простая и гладкая модель.

Выбор λ:

- **Слишком маленький** λ: слабая регуляризация → риск переобучения
- **Слишком большой** λ: сильная регуляризация → все веса около нуля → недообучение
- **Оптимальный** λ: подбирается по валидационной выборке

4. Записать функционал Lasso-регрессии (L1-регуляризации) и объяснить, почему L1-штраф часто приводит к появлению нулевых коэффициентов (разреженности модели).

Функционал Lasso-регрессии

Задача: минимизировать J_{lasso}

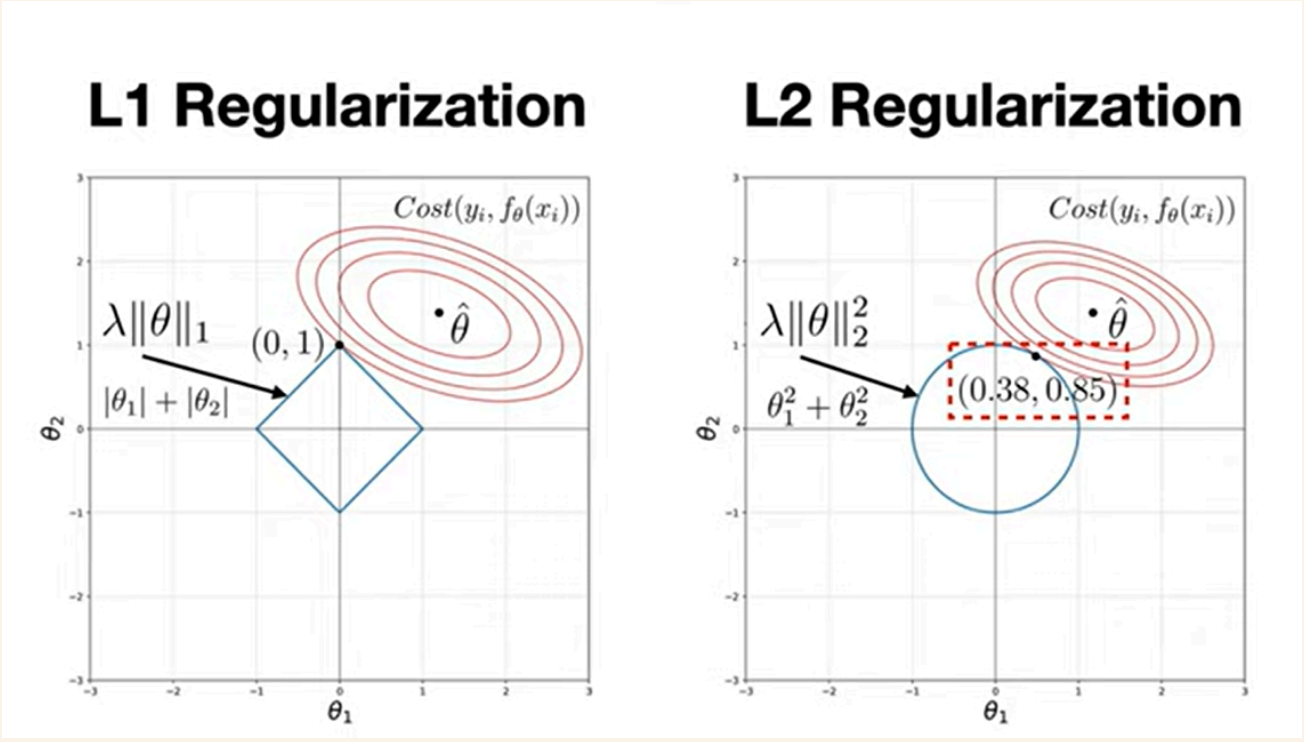
$$J_{\text{lasso}}(w, b) = \text{MSE}(w, b) + \lambda \|w\|_1 = \frac{1}{n} \sum_{i=1}^{n} (y_i - (w^\top x_i + b))^2 + \lambda \sum_{j=1}^{d} |w_j|$$

где $\|w\|_1$ — L1-норма (сумма абсолютных значений весов).
λ - штраф даёт такие же эффекты, как и в пунктах 2-4 L2.

Разреженность модели

L1-штраф часто приводит к обнулению некоторых коэффициентов, создавая разреженную модель.

Геометрическая интерпретация:



Синее - штраф; Красное - MSE; Черная точка - решение

- **L2 (Ridge)**: Ограничение в виде сферы → решение часто находится на гладкой поверхности.
- **L1 (Lasso)**: Ограничение в виде ромба → решение часто попадает в углы, где некоторые координаты (веса) равны нулю.

Практический смысл: Lasso выполняет автоматический **отбор признаков** — неважные признаки получают нулевой вес.

5. Сравнить Ridge и Lasso-регрессии: в каких ситуациях предпочтителен каждый из методов, что такое Elastic Net и как он сочетает L1 и L2-штрафы.

Сравнительная таблица

Критерий	Ridge (L2)	Lasso (L1)
Функционал	$\text{MSE}(\text{w}, b) + \lambda \|\text{w}\|_2^2$	$\text{MSE}(\text{w}, b) + \lambda \|\text{w}\|_1$
Влияние на веса	Сжимает к нулю, но не обнуляет	Может обнулять некоторые веса(разреженность)
Мультиколлинеарность	Делит вес между коррелирующими признаками	Выбирает один признак среди коррелирующих
Интерпретируемость	Средняя (все признаки остаются)	Высокая (остаются только важные признаки)
Когда предпочтительнее	Много слабых признаков, важна устойчивость	Мало важных признаков, нужен отбор признаков, важна интерпретируемость

Elastic Net

Комбинирует L1 и L2 регуляризацию:

$$J_{\text{elastic}}(w, b) = \text{MSE}(w, b) + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$$

$\lambda_1 = 0$ - применяем только Ridge,
$\lambda_2 = 0$ - применяем только Lasso

- **Преимущества**: Сочетает отбор признаков (L1) и устойчивость к мультиколлинеарности (L2), уменьшает переобучение
- **Применение**: Когда много признаков и среди них есть корреляции.

6. Объяснить, зачем перед применением регуляризованных моделей стандартизуют признаки. Как на практике подбирают параметр регуляризации?

Необходимость стандартизации

Регуляризованные модели **чувствительны к масштабу признаков**, так как штраф применяется ко всем весам одинаково.

> ***Проблема без стандартизации***:
>
> - Признак с большим размахом значений (например, зарплата: 50 000-200 000) будет иметь меньший вес.
> - Признак с малым размахом (например, возраст: 18-65) будет иметь больший вес.
> - L1/L2 штраф "несправедливо" наказывает веса, не учитывая масштаб признаков.

Стандартизация решает:

> 1. **Справедливый штраф**: Все признаки приводятся к одинаковому масштабу.
> 2. **Корректная регуляризация**: Штраф применяется равномерно ко всем признакам.
> 3. **Ускорение сходимости**: Градиентный спуск работает эффективнее.

Методы масштабирования:

> - **StandardScaler**: $x_{\text{scaled}} = \frac{x-\mu}{\sigma}$
>
> μ - среднее по признаку, σ - стандартное отклонение
> удаляет среднее значение и масштабирует данные до единичной дисперсии
>
> - **MinMaxScaler**: $x_{\text{scaled}} = \frac{x-x_{\min}}{x_{\max}-x_{\min}}$
>
> масштабирует все признаки данных в диапазоне $[0, 1]$

Подбор параметра λ

> 1. **Кросс-валидация**: Разбиение данных на k фолдов, обучение на k-1, тест на 1.
> 2. **Стандартизация**
> 3. **Кривые валидации**: Построение графика ошибки на обучении и валидации от λ . Проверка на пере/недообучение.
> 4. **Выбор оптимального λ**, где ошибка на валидации минимальна.
> 5. **Сравнение моделей**: Попробовать Ridge, Lasso, Elastic Net на валидации.
> Еще можно использовать поиск по сетке GridSearch, заменив шаги 2, 3 4.

Лекция 3. Логистическая регрессия

> **1. Сформулировать задачу бинарной классификации в логистической регрессии. Что считается входом и выходом модели?**

Задача бинарной классификации

> Логистическая регрессия решает **задачу бинарной классификации**. Цель — по объекту, представленному вектором признаков, предсказать, к какому из двух классов (0 или 1, Negative или Positive) он принадлежит. Модель предсказывает не метку класса напрямую, а ***вероятность*** принадлежности объекта к целевому классу.

Вход и выход модели

> **Вход модели**:
>
> - Вектор признаков объекта $x = [w_1, w_2, \dots, w_n]^\top$ где x_j — веществennозначные или бинарные признаки.
> - По сути, тот же вход, что и для линейной регрессии.
>
> **Выход модели**:
>
> - **Вероятность** ($\hat{p}^{(i)} = P(y^{(i)} = 1 \mid x^{(i)})$), то есть вероятность того, что объект (i) принадлежит классу 1 при заданных его признаках.
> - Для получения этой вероятности линейная комбинация признаков ($z = w^\top x + b$) пропускается через **сигмоидную (логистическую) функцию** (\sigma(z)).

Математическая формулировка

> 1. **Линейный слой (логит)**: ($z^{(i)} = w^\top x^{(i)} + b$)
> 2. **Активация (сигмоида)**: ($\hat{p}^{(i)} = \sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$)
>
> **Расширенная форма** (с включением (b) в вектор весов):
>
> $$\hat{p}^{(i)} = \sigma(\tilde{w}^\top \tilde{x}^{(i)}) = \frac{1}{1 + e^{-\tilde{w}^\top \tilde{x}^{(i)}}}$$
>
> где ($\tilde{x}^{(i)} = [1, x_1^{(i)}, \dots, x_n^{(i)}]^\top$, $\tilde{w} = [b, w_1, \dots, w_n]^\top$)

Получение итогового класса

> Вероятность ($\hat{p}^{(i)}$) является основным выходом модели. Итоговая метка класса ($\hat{y}^{(i)}$) получается применением **порога** (по умолчанию 0.5):
>
> $$\hat{y}^{(i)} = \begin{cases} 1, & \text{если } \hat{p}^{(i)} \geq 0.5 \\ 0, & \text{если } \hat{p}^{(i)} < 0.5 \end{cases}$$

2. Объяснить, почему обычная линейная регрессия плохо подходит для решения задач бинарной классификации

1. Линейная регрессия выдает любые числа, а требуется вероятность от 0 до 1
2. Нерелевантность квадратичной ошибки: за очень уверенную ошибку штраф почти как за небольшую
3. Нужна модель, у которой $\hat{p} = Pr(y = 1 \mid x)$, и потери, сильно наказывающие "уверенно неправые" ответы

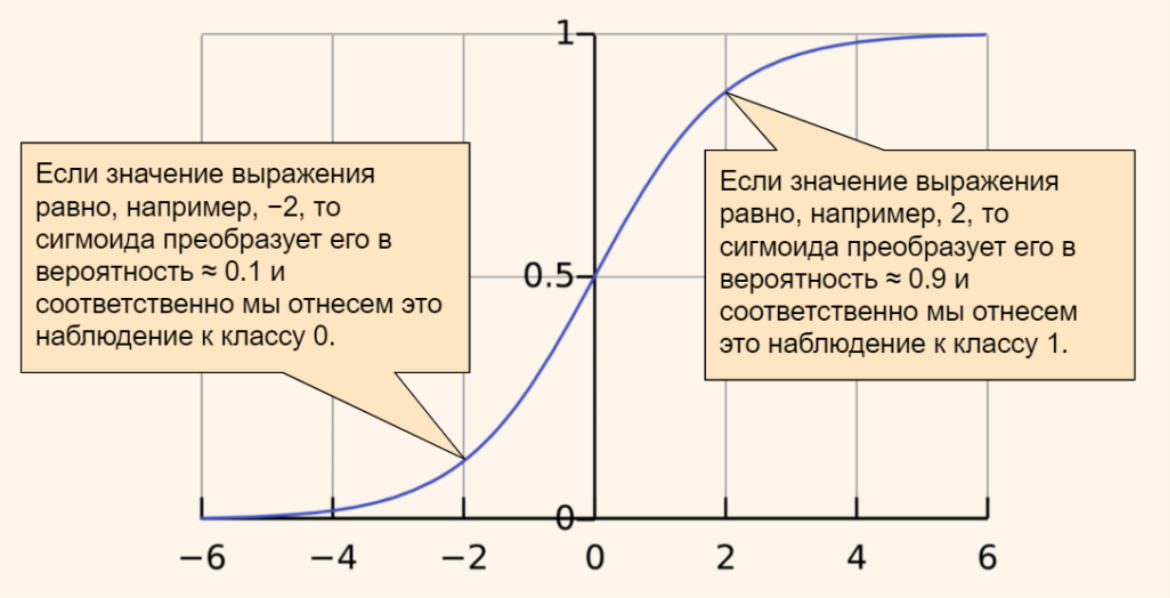
3. Записать и_ли объяснить выражение для вероятности наблюдения y є {0,1} при фиксированном x и параметрах модели. Записать и_ли объяснить правдоподобие выборки как произведение по объектам[1].

Вероятность в логистической регрессии

В логистической регрессии используется распределение Бернулли с параметром $p = \sigma(z)$, где:

- $\sigma(z) = \frac{1}{1+e^{-z}} \in [0, 1]$ — сигмоидная функция
- $z = w^\top x + b$ — линейный отклик

  Сигмоида задает параметр распредления Бернулли, и в отличие от линейной регрессии, на выход выдается вероятность. Чем дальше от границы решений ($z = 0$) , тем выше уверенность.



По оси абсцисс - z, по оси ординат - $\sigma(z)$

Вероятность для одного объекта:

$$p(y \mid x) = \sigma(z)^y \cdot (1 - \sigma(z))^{(1-y)}, \quad y \in \{0, 1\}$$

Объяснение формулы:

- y - правильный ответ
- $\sigma(z)$ - вероятность класса 1
- $(1 - \sigma(z))$ - вероятность класса 0
- Степени y и $(1 - y)$ "включают" или"выключают" множитель.

- Если $y = 1$: $p(1 \mid x) = \sigma(z)^1 \cdot (1 - \sigma(z))^0 = \sigma(z)$
- Если $y = 0$: $p(0 \mid x) = \sigma(z)^0 \cdot (1 - \sigma(z))^1 = 1 - \sigma(z)$

Таким образом, формула описывает вероятность для обоих классов.

Правдоподобие всей выборки

При предположении независимости объектов:

$$\mathrm{L}(w, b) = \prod_{i=1}^{n} \sigma(z_i)^{y_i} \cdot (1 - \sigma(z_i))^{(1-y_i)}$$

где $z_i = w^\top x_i + b$.
Это произведение вероятностей по всем объектам выборки. Это значение мы хотим максимизировать.

4. Вывести, привести и_ли объяснить выражение для функции потерь логистической регрессии (log loss)[1].

Вывод log loss (кросс-энтропийная потеря) из правдоподобия

1. Логарифмирование правдоподобия (удобнее работать с суммой):

- после перемножения чисел от 0 до 1 итоговое произведение будет очень маленьким (при большой выборке)
- работать с суммами гораздо удобнее чем с произведением.
- суммы лучше дифференцируются, эффективный градиентный спуск
- y_i, которое стояло в степени, выносится по свойству логарифмов.

$$\log \mathrm{L}(w, b) = \sum_{i=1}^{n} [y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i))]$$

2. Преобразование к функции потерь:

- Меняем знак (максимизация правдоподобия → минимизация потерь)
- Добавляем усреднение по выборке:

$$J(w,b) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))\right]$$

Log loss мы хотим минимизировать

5. Пояснить, как и зачем добавляются L1- и L2-регуляризации в логистическую регрессию. Как регуляризация влияет на переобучение и интерпретируемость модели?

Регуляризация в логистической регрессии

Аналогично линейной регрессии, в логистическую регрессию добавляются штрафы на веса для борьбы с переобучением. См. 2.3-6

Регуляризованные функционалы

L2-регуляризация (Ridge):

$$J_{\text{ridge}}(w,b) = J(w,b) + \lambda \|w\|_2^2$$

L1-регуляризация (Lasso):

$$J_{\text{lasso}}(w,b) = J(w,b) + \lambda \|w\|_1$$

где $J(w,b)$ — исходный log loss, $\lambda > 0$ — коэффициент регуляризации.

Влияние регуляризации

Аспект	L2 (Ridge)	L1 (Lasso)
Переобучение	Уменьшает, "сжимая" все веса	Уменьшает, обнуляя неважные веса
Интерпретируемость	Все признаки остаются, сложнее интерпретировать	Автоматический отбор признаков, проще интерпретировать
Устойчивость	Устойчивость к мультиколлинеарности	Выбирает один признак из коррелирующих
Разреженность	Нет (веса маленькие, но не нулевые)	Да (многие веса точно равны 0)

Выбор типа регуляризации

- L2 (Ridge): когда важны все признаки, но нужно контролировать их влияние.
- L1 (Lasso): когда нужно выбрать наиболее важные признаки для интерпретации.

Лекция 4. Решающие деревья и ансамбли (бэггинг, случайные леса, стэкинг)

1. Описать структуру решающего дерева: что такое внутренний узел, лист, ребра, глубина дерева. Чем дерево классификации отличается от дерева регрессии по типу предсказания в листе?

- **Внутренний узел**- узел с условием, по которому делается предсказание
- **Лист** - предсказание дерева, может быть классом (классификация) или числом (регрессия)
- **Ребро** - варианты ответа на условие, заданное в листе (true/false)
- **Глубина дерева** - максимальное количество вопросов от корня до листа.

У дерева классификации в листе находится класс, у дерева регрессии - число.

2. Объяснить пошагово, как решающее дерево делает предсказание для одного объекта.

Начинаем с корня дерева. Если для объекта условие выполняется, идем по ребру “да”, иначе - по ребру “нет”. После этого мы можем попасть либо во внутренний узел с условием, либо в лист с предсказанием. Если внутренний узел - повторяем то же, что и с корнем. Если лист с предсказанием - выдаем предсказание.

3. Описать жадный алгоритм построения дерева сверху вниз. Какие критерии останова используются (максимальная глубина, минимальное число объектов в листе, минимальное уменьшение нечистоты)?

Пусть в узле выборка S с долями классов $p_1, \dots, p_K$

Индекс Джини

$$Gini(S) = 1 - \sum_{k=1}^{K} p_k^2$$

Энтропия"

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

Обучение

Для выборки ищем такие признак и порог , для которых при разбиении на две подвыборки разные классы будут максимально (в этом и заключается жадность алгоритма - ищем максимум) распределены по подвыборкам (можно замерить индексом Джини, или посчитав энтропию). Выбранное условие ставится в узел, затем то же самое проводится с листами получившегося узла.

Когда достигаем максимальной глубины, в лист ставим предсказание, может быть классом (классификация) или числом (регрессия) - мода для подвыборки.

- **Максимальная глубина** - гиперпараметр, на какой глубине ставить лист. Нужен, чтобы избежать переобучения. При маленьких значениях модель обучится плохо, при больших - переобучится.
- **Минимальное число объектов в листе** - гиперпараметр, при скольки объектах нужно ставить лист. Нужен, чтобы избежать переобучения. Если объектов в подвыборке мало, то ставится лист. При маленьких значениях модель обучится плохо, при больших - переобучится.
- **Минимальное уменьшение нечистоты** - гиперпараметр, если для условия изменение энтропии или индекса Джини мало, вместо условия ставится лист. Опять же, для избежания переобучения.

4. Объяснить, как деревья работают с числовыми и категориальными признаками, как можно обрабатывать пропуски и почему деревья обычно не требуют масштабирования признаков.

Работа с признаками

- Числовые - ищем порог с максимальной энтропией. Например, возраст > x, возраст < x
- Категориальные - бинарные сплиты, по сути разделение на “принадлежит категории”/”не принадлежит категории”. Например, красные машины, не красные машины.

Пропуски

Ставим моду (самое частое значение) или медиану, еще можно использовать Surrogate split - при прохождении узла идем туда, где большая подвыборка.

Масштабирование

Масштабирование не требуется, т.к. основное действие - сравнение с порогом, поэтому нет разницы, как будут распределены признаки и какие у них значения.

Могут потребоваться, если дерево стакается с другой моделью, или если данные настолько кривые, что ломают индекс Джини или энтропию.

5. Рассказать о причинах переобучения решающих деревьев и способах борьбы с ним: пред-обрезка и пост-обрезка.

Переобучение в дереве возникает, если в листе оказывается 1 или малое кол-во объектов.

- **Пред-обрезка** - обрезка при помощи гиперпараметров во время построения дерева, см. вопрос 3
- **Пост-обрезка** - обрезаем после построения дерева те листы или узлы, которые не сильно улучшают качество

6. Перечислить основные плюсы и минусы решающих деревьев как модели для табличных данных.

> **Плюсы**

- Интерпретируемы (можно посмотреть, как проводилось предсказание)
- Не требуют масштабирования (см вопрос 4)
- Умеют смешивать числовые и категориальные признаки
- Быстро работают (сложность - O(n), где n - глубина)

Минусы

- Склонность к переобучению, если не принять меры
- Кусочно-постоянное предсказание (выдает не ровный график, а ступеньки)
- Нестабильность при малых изм. данных

7. Объяснить идею бэггинга (Bootstrap Aggregating): как формируются бутстрап-выборки, как усредняются предсказания, что такое OOB-оценка качества.

Основная идея бэггинга

Обучение нескольких слабых моделей, которые будут "голосовать за результат". Для того, чтобы они были немного разными, для каждой модели (дерева) в лесу формируется своя бутстрап-выборка (случайная выборка объектов из обучающей выборки)

Выбор предсказания

- Для классификации - за что проголосовало большинство (или можно шанс принадлежности к классу выдать),
- Для регрессии - среднее между предсказаниями

OOB-оценка

Для каждого дерева найдутся объекты, не попавшие в его бутстрап-выборку, по ним можно быстро оценить точность дерева (и репрессировать, если точно мала)

8. Дать определение случайного леса. Чем он отличается от обычного бэггинга деревьев и какую роль играет ограничение числа признаков при поиске сплита (параметр max_features)?

Случайный лес - тот же бэггинг деревьев, но для каждого сплита (выбора условия) используются не все, а случайные признаки, из-за чего каждое дерево отличается от собрата еще сильнее, что делает ансамбль сильнее.

Max features - гиперпараметр, из скольки признаков будет выбран на сплит.

9. Объяснить идею стэкинга: что такое базовые модели и мета-модель, как формируются признаки второго уровня. Что такое OOF-предсказания и зачем они используются?

Идея стэкинга

Обучение нескольких моделей, а затем обучение мета-модели на предсказаниях базовых моделей (признаки второго уровня), чтобы научиться их комбинировать лучше среднего/моды.

Знания про OOF-предсказания, по словам ВИ, на зачете не нужны (но вообще это что-то вроде OOB - оценки для стэкинга)

Лекция 5. Бустинг и градиентный бустинг над деревьями

1. Сформулировать общую идею бустинга. Чем бустинг принципиально отличается от бэггинга и стэкинга?

Общая идея

Последовательное добавление слабых моделей, каждая из которых исправляет ошибки предыдущей, i.e. предсказывает ошибку предыдущей.

Отличие от бэггинга и стэкинга

При бэггинге обучется несколько моделей, и финальное предсказание формируется "голосованием" (берется мода или среднее от всех предсказаний)
При стэкинге на предсказаниях нескольких моделей обучается еще одна модель, которая сможет лучше комбинировать их ответы, чем просто мода или среднее.
При бустинге же модели работают как бы не параллельно, а последовательно. Каждая последующая модель, кроме первой, старается предсказать не таргет переменную, как в бэггинге или стэкинге, а ошибку предыдущей модели.
Бэггинг и стэкинг менее чувствительны к отдельным выбросам. Бустинг чувствителен к шуму (важна регуляризация).

2. Записать и_ли объяснить объяснить общее обновление модели в градиентном бустинге [1]

$$F_M(x) = F_{M-1}(x) + v * h_m(x)$$

Где

- $F_0(x)$ - изначальная модель, которая грубо предсказывает ошибку
- $h_M(x)$ - модель, которая предсказывает ошибки $F_{M-1}(x)$
- $v \in (0; 1]$ - скорость обучения (i.e. как сильно модель каждой итерации влияет на конечную модель. Чем меньше скорость обучения, тем больше нужно обучить моделей, но тем более точно можно скорректировать финальную модель)

Оно же:

$$F(x) = F_0(x) + \sum_{i=1}^{h} h_i(x) * v^i$$

3. Пояснить точку зрения градиентного спуска в пространстве функций: как задается функционал ошибки L(F) и какую роль играют псевдо-остатки (антиградиенты) на обучающих объектах.

Пространство функций

Пространство функций - это множество слабых функций h, которые мы перебираем для нахождения лучшей финальной функции F

Как задается функционал ошибка L(F)

$$L(F) = \sum_{i=1}^{n} l(y_i, F(x_i))$$

Где F - искомая функция, а l - выбранная функция потерь.

Псевдо-остатки

На каждом шаге нам нужна модель h, которая будет предсказывать ошибку прошлой модели F.

$$g_{im} = \frac{\partial l(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}$$

$$h_m \approx -g_{im}$$

I.e. псевдо-остатки (производная Фреше в обобщенном метрическом пространстве) пропорциональны предсказаниям модели h.

Градиентный бустинг называется так, т.к. формула сильно похожа на формулу градиентного спуска

$$x = x_0 - \sum_{i=1}^{n} v_i g'(x_i)$$

4. Записать и_ли объяснить шаг алгоритма градиентного бустинга над деревьями: вычисление псевдо-остатков, обучение регрессионного дерева по этим значениям, поиск оптимальных сдвигов по листам и обновление ансамбля [1]

1. Вычисление псевдо-остатков

$$r_{im} = \frac{\partial l(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}$$

2. Обучение регрессионного дерева $h_m(x)$ по парам (x_i, r_{im})
3. Нахождение оптимального сдвига (константы) по листам: для каждого листа R_{jm}

$$\gamma_{jm} = arg\min_{\gamma} \sum_{x_i \in R_{jm}} l(y_i, F_{m-1}(x_i) + \gamma)$$

4. Обновление модели

$$F_m(x) = F_{m-1}(x) + v * \sum_j \gamma_{jm} 1\{x \in R_{jm}\}$$

5. Перечислить основные способы регуляризации в градиентном бустинге: малая скорость обучения v, ограничение глубины деревьев, минимальный размер листа, субсемплинг объектов, субсемплинг признаков, ранняя остановка по валидационной выборке.

1. Скорость обучения (shrinkage) $v \in (0; 1]$. Чем меньше, тем устойчивее, но нужно больше деревьев.
2. Глубина дерева / число листьев: неглубокие деревья (3-8 уровней) приводят к слабым базовым моделям.
3. Субсемплинг объектов: обучение h_m на случайной доле данных (обычно от 0.5 до 0.9)
4. Субсемплинг признаков: случайный поднабор признаков на сплите/уровне/дереве.
5. Минимальный размер листа, L2-штраф на веса листок, макс. число узлов.
6. Ранняя остановка по валидации: мониторинг метрики и прекращение роста M (перетекает в некст вопрос)

6. Объяснить, как по поведению метрик на обучающей и валидационной выборках диагностировать переобучение бустинговой модели и как выбирать оптимальное число итераций.

Переобучение

При подборе гиперпараметров нужно стремиться к уменьшению метрики на валидации и сохранению метрики на обучающей выборке. Если материка на обучающей выборке уменьшается, а на валидационной - нет, то это признак переобучения.

Число итераций

Подбирается так же, как и любой гиперпараметр, - по метрике валидационной выборке

Лекция 6. Кластеризация, SVD и PCA

1. Дать определение обучению без учителя. Какие задачи обычно относят к обучению без учителя? Приведите примеры.

Определение

Обучение без учителя - это метод машинного обучения, при котором алгоритм работает с неразмеченными данными (отсутствие таргет-переменной, y) с целью обнаружить, как организованы данные.

Примеры

Примерами являются: понижение размерности (матриц из данных) и кластеризация (разбиение данных на кластеры)

Примеры в реальной жизни: Геологическое обнаружение минералов, кластеризация генов, машинное зрение.

2. Описать алгоритм K-Means. Записать и объяснить целевую функцию, которую алгоритм минимизирует

Алгоритм

1. Выбирается K случайных (или по какому-то алгоритму) стартовых точек (центроидов)
2. Каждый элемент данных относится к тому же “кластеру”, что и ближайшая стартовая точка
3. В каждом созданном кластере выбирается самый средний элемент - он будет выступать новым центроидом. Для новых центроидов снова выполняем пункт 2.
4. Повторяем пункты 2-3 до тех пор, пока кластеры практически не перестанут менятся

> ### Минимизируемая функция
>
> $$J(C, \mu) = \sum_{k=1}^{K} \sum_{\mathrm{x}_i \in C_k} \|\mathrm{x}_i - \mu_k\|^2$$
>
> где
>
> - x_i - точка
> - $\mu_{c(i)}$ - центр ее кластера
>
> То есть сумма от сумм расстояний каждого объекта до центроида для каждого кластера. Все расстояния суммируем для кластера а потом и все кластеры суммируем

3. Перечислить основные преимущества и недостатки K-Means. Как влияет масштабирование признаков и наличие выбросов на работу алгоритма?

Плюсы

1. Быстрый (по сравнению с двумя другими известными алгоритмами) - асимптотическая сложность ниже ($n * k * d * t$ против кубических и квадратных), при этом операции вычисления оптимизированы под устройство процессора (векторные регистры)
   2. Масштабируемость на размеры - $O(n) = n * k * d * t$, где k - количество центроидов, d - длина векторов (features), t - количество итераций
2. Интерпретируемый - результатом становятся вполне объяснимые центры и близкие к ним данные

Минусы

1. Немасштабированные данные сильно ухудшают работоспособность, так как признаки с большим разбросом данных (большой масштаб) будут важнее более плотных
2. Выбросы также ломают модель, так как минимизируемая функция - квадрат расстояний, соответственно новый центроид будет сильно сдвинут

4. Объяснить алгоритм DBSCAN. Каковы роли параметров e (epsilon) и min_samples? Чем отличаются ядровые, пограничные и шумовые точки?

> #### Идея
> Кластерами будут зоны “высокой плотности” объектов, которые разделены границами - зонами “низкой плотности”.

> #### Алгоритм
>
> 1. Выбираем произвольную точку.
> 2. Берём всех соседей в e (epsilon)-окрестности.
> 3. Если количество соседей >= min_samples - в этой точке считаем новый кластер.
> 4. Если соседей меньше min_samples или точка не принадлежит уже созданному кластеру, то точка помечается как шум
> 5. Проходим также все оставшиеся точки.

> #### Ядровые, пограничные и шумовые точки

> Ядровые - принадлежащие “внутренней” части обнаруженного кластера, не участвуют в обновлении новых точек.
>
> Пограничные - точки на “границе” кластера - через них и epsilon и высчитывается принадлежность следующих точек.
>
> Шумовые - помеченные как шум.

5. Сравнить DBSCAN и K-Means по форме находящихся кластеров, устойчивости к выбросам, необходимости заранее задавать число кластеров и чувствительности к плотности данных?

Форма

- K-Means - n-мерная сфера (окружность, при двух признаках), так как принадлежность определяется расстоянием до центра (центроида)
- DBSCAN - произвольная форма, так как “расширение” кластера происходит засчёт граничных точек (ближайшие к границе)

Устойчивость к выбросам

- K-Means - отсутствует, так как выбросы сильно смещают центроиды (см. выше про квадрат расстояний)
- DBSCAN - Высокая, так как способен различать шумы

Число кластеров

- K-Means: заданное, K
- DBSCAN: сам определяет их

Чувствительность к плотности данных

- K-Means: он игнорирует локальную плотность, разделяет зоны на равные по удалённости
- DBSCAN: алгоритм основан на плотности, а потому равноплотные данные ухудшают работоспособность

6. Описать идею агломеративной иерархической кластеризации: начальное состояние, правило слияния кластеров, возможные варианты связи (single, complete, average, Ward), роль дендрограммы.

> #### Идея
> Не разделять данные на кластера сразу, а объединять близкие малые кластеры для

получения больших. Принцип “снизу-вверх”.

Начальное состояние

Каждый объект из выборки - кластер (всего их n)

Правило слияния

Выбираем у каждого кластера ближайший в соответствии с выбранным вариантом связи, объединяем их. Затем повторяем слияние новых кластеров с пересчитанным расстоянием

Варианты связи

- Single - расстояние между ближайшими точками кластеров
- Complete - между самыми далёкими элементами
- Average - между средними, “центрами” кластеров
- Ward - сложный вид, на доп вопрос (кратко - минимизация внутрикластерной дисперсии)

Дендрограмма

Дендрограмма - древовидная диаграмма, которая визуализирует слияние малых кластеров (листьев) в конечные кластера. Уровень, на котором будет сделан “срез” диаграммы отсекает количество итоговых кластеров.

7. Перечислить достоинства и недостатки иерархической кластеризации, в том числе с точки зрения вычислительной сложности и работы с большими выборками

Достоинства

1. Интерпретируемость. Наглядно показывает вложенность данных внутри кластера
2. Не требует задавать число кластеров, однако необходимо выбрать уровень детализации (уровня отсечения разных кластеров)
3. Результат воспроизводимый, так как объединение кластеров не зависит от “случайности”

Недостатки

1. Затрачивает много памяти (необходимо хранить расстояние между всеми парами значений)
2. Чувствителен к шуму. Объединение с близким шумовым объектом на раннем этапе ведёт к “связыванию” кластера с шумом

3. Необходимо выбирать метод связи - от этого зависит результат.
4. Не учитывает, что данные могут быть разбросаны произвольно, а не сферически-подобно

Особенные недостатки

5. Как и с памятью, необходимо делать очень много вычислений $O(n) = n^3$ => долго
6. Из-за такой высокой асимптотической сложности, физически не способен кластеризировать выборки размером более 10^6 (или больше)

8. Сформулировать основные цели снижения размерности и назвать несколько наиболее популярных алгоритмов.

Цели

- Убрать мультиколениарные (говорящие об одном и том же, коррелирующие) признаки
- Избавится от шумов (избавление от случайных значений
- Решает проблему проклятия размерности (чем больше фич - тем труднее модели обучатся и легче переобучаться)
- Ускорение работы моделей, обучаемых на данных: меньше фич -> меньше вычислений
- Позволяет визуализировать данные. Например, в рамках кластеризации помогает преобразовать многомерное пространство в 2/3-мерное (чтобы наглядно увидеть распределение данных)
- Сжимает данные -> меньше памяти
- Выявление скрытых (латентных) признаков

Популярные алгоритмы

(в подробностях - на доп вопрос, помните название только)

- PCA (Principal Component Analysis) - суть: преобразовать в новые оси (из многомерности в n-мерность), максимизируя при этом дисперсию (чтобы значение признака было говорящим)
- SVD (Singular Value Decomposition) - матричная факторизация. Буквально поворот матрицы и умножение на саму себя для сжатие размерности (матрица размера $n * m$ становится матрицей размера $\min(n, m)^2$)

- t-SNE (t-distributed stochastic neighbour embedding) - один из нелинейных методов снижения размерности