

Вопросы для экзамена 2025-2026 по ML

29 декабря 2025 г.

Содержание

- | | | |
|----|---|----|
| 1 | Опишите задачу классификации. Что такое матрица ошибок? Что показывает accuracy? Чем плох этот показатель? | 4 |
| 2 | Что такое ошибки I и II рода? Что показывает ROC–кривая, как она выглядит, какой числовой показатель качества модели используется? | 5 |
| 3 | Что такое recall и precision? Как устроена PR–кривая и что она показывается? Что такое F_1 и F_β –меры, зачем они нужны? | 6 |
| 4 | Выполните формулу байесовского классификатора. Докажите, что он имеет наибольшую точность. | 7 |
| 5 | Опишите наивный байесовский подход. Опишите классификаторы LDA и QDA. Выполните формулы LDA и QDA методом наибольшего правдоподобия. | 7 |
| 6 | Опишите логистический классификатор. Дайте определение KL–дивергенции. Выполните с помощью нее функцию потерь классификатора. | 8 |
| 7 | Опишите алгоритм градиентного спуска на примере логистического классификатора. Проверьте функцию потерь на выпуклость. | 9 |
| 8 | Опишите задачу многоклассовой классификации и метод softmax regression. Выполните функцию потерь из метода наибольшего правдоподобия. | 10 |
| 9 | Поставьте задачу кластеризации, приведите примеры разных метрик. Дайте определения метрик качества: внутри- и межкластерное расстояние, WSS, BSS, Silhouette. | 11 |
| 10 | Опишите методы k–means и k–medoids. Приведите алгоритмы решения Lloyd и Elkan. Оцените их скорость. | 12 |
| 11 | Выполните EM–algorithm кластеризации для смеси нормальных распределений. | 13 |

12 Опишите иерархические методы: ближнего соседа, дальнего соседа, средней связи, центроидов, Уорда. Выпишите формулу Ланса–Вильямса.	13
13 Опишите алгоритм DBScan. Опишите алгоритм OPTICS.	14
14 Сформулируйте алгоритм спектральной кластеризации. Выведите его на примере двух классов.	15
15 Опишите алгоритм понижения размерности. Выведите формулу для PCA.	16
16 Выведите формулу для вероятностного PCA. Как выбирать число измерений проекции?	16
17 Опишите нелинейный алгоритм PCA. Приведите примеры разных ядер.	17
18 Дайте постановку метода SVC с мягким и жестким зазором. Сведите обе задачи к задачам условной оптимизации.	18
19 Сформулируйте теорему Куна–Таккера. Сведите задачу SVC с жестким зазором к двойственной задаче.	18
20 Опишите и выведите нелинейный метод SVC для задачи с жестким зазором.	19
21 Сформулируйте теорему Мерсера. Приведите примеры ядер для нелинейного SVC.	20
22 Сведите задачу многоклассовой классификации к бинарной методами one–versus–one и one–versus–all.	21
23 Дайте постановку метода SVR. Сведите задачи с мягким и жестким зазором к задачам условной оптимизации.	21
24 Сформулируйте теорему Куна–Таккера. Выведите двойственную задачу для метода SVR с мягким зазором. Опишите ядерный метод SVR.	22
25 Опишите метод KNN. Приведите примеры различных метрик. Что такое весовой KNN?	23
26 Опишите метод скользящего окна для метрической классификации и регрессии. Приведите примеры различных ядер.	24
27 Дайте математическое (вероятностное) обоснование методов KNN и скользящего окна классификации и регрессии.	25
28 Выведите формулы локальной регрессии из МНК. Сравните локально–постоянную и локально–линейную регрессии.	26

29 Опишите деревья принятия решений, опишите алгоритм использования. Опишите алгоритм построения дерева, оцените его сложность.	27
30 Дайте определение информативности. Выведите ее формулы из минимизации функции потерь — абсолютной, бинарной, квадратичной, логарифма правдоподобия.	28
31 Выведите формулу Bias–Variance trade–off для задачи регрессии. Покажите уменьшение дисперсии при применении пастинга.	29
32 Опишите метод бэггинга. Приведите аргументы в пользу его обоснования.	30
33 Опишите алгоритм построения случайного леса. Как выбирать параметры леса?	31
34 Опишите алгоритмы стекинга и блэндинга. Как находить и использовать feature importance?	32
35 Опишите алгоритм градиентного бустинга.	33
36 Выведите формулы градиентного бустинга на решающих деревьях для квадратичной и логистической функции потерь. Приведите примеры методов ускорения бустинга.	33
37 Опишите перцептрон Розенблатта. Приведите примеры других функций активации.	34
38 Дайте определение нейронной сети. Сформулируйте теорему Цыбенко.	35
39 Выведите формулы метода обратного распространения ошибок на примере сети с одним скрытым слоем. Сформулируйте общий принцип.	35
40 Опишите способы инициализации Kсавье и Kaiming. Приведите их математический вывод.	36
41 Опишите прием Dropout. Дайте его математическое обоснование.	37
42 Опишите приемы изменения скорости обучения и нормализации для обучения нейросетей.	38

1 Опишите задачу классификации. Что такое матрица ошибок? Что показывает accuracy? Чем плох этот показатель?

Задача классификации (обучение с учителем): по обучающей выборке $D = \{(\vec{x}_i, y_i)\}_{i=1}^n$, где $\vec{x}_i \in \mathbb{R}^m$ — вектор признаков, а y_i — метка класса, построить классификатор $g(\vec{x})$, который для новых объектов выдаёт предсказание класса.

- **Бинарная классификация:** $y \in \{0, 1\}$ (классы C_0 и C_1).
- **Многоклассовая классификация:** $y \in \{1, \dots, K\}$.

Практическая цель ML-формулировки — хорошо классифицировать новые данные, т.е. минимизировать долю ошибок на новых наблюдениях (обобщающая способность), а не «идеально объяснить» обучающую выборку.

Матрица ошибок (confusion matrix, матрица неточностей): таблица счётов, где строки соответствуют истинным классам, а столбцы — предсказанным. Элемент C_{ij} показывает число объектов истинного класса i , отнесённых моделью к классу j .

Для бинарной классификации удобно вводить четыре числа:

- TP (true positive): $y = 1, \hat{y} = 1$;
- FN (false negative): $y = 1, \hat{y} = 0$;
- TN (true negative): $y = 0, \hat{y} = 0$;
- FP (false positive): $y = 0, \hat{y} = 1$.

Тогда матрицу ошибок можно записать как

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}.$$

Accuracy (верность классификации): доля верных ответов модели на выборке. В бинарном случае

$$\text{Acc} = \frac{TP + TN}{n}, \quad n = TP + TN + FP + FN.$$

В многоклассовом случае

$$\text{Acc} = \frac{\sum_{k=1}^K C_{kk}}{\sum_{i=1}^K \sum_{j=1}^K C_{ij}} = \frac{\text{trace}(C)}{n}.$$

Интерпретация: это эмпирическая вероятность того, что классификатор угадал метку.

Чем плох accuracy: если классы несбалансированы ($n_0 \neq n_1$), метрика может «врать». Например, пусть класс C_0 встречается в 95% случаев, а C_1 — в 5%. «Глупая» модель $\hat{y} \equiv 0$ имеет $\text{Acc} = 0.95$, хотя для класса C_1 у неё $TP = 0$ (то есть она никогда не находит редкий класс). Кроме того, accuracy не учитывает разную цену ошибок: в разных задачах критичнее либо пропустить объект класса C_1 (ошибка I рода / FN), либо сделать ложную тревогу (ошибка II рода / FP). Поэтому при дисбалансе классов и/или неравной стоимости ошибок обычно переходят к метрикам, зависящим от TP, FN, FP, TN (recall, precision, ROC/PR и т.д.).

2 Что такое ошибки I и II рода? Что показывает ROC–кривая, как она выглядит, какой числовой показатель качества модели используется?

Ошибки I и II рода (в терминах лекций): пусть в бинарной классификации мы следим за классом C_1 (*positive class*), а C_0 — отрицательный. Тогда

- **ошибка I рода:** модель сказала «нет» ($\hat{y} = 0$), хотя на самом деле «да» ($y = 1$), т.е. **false negative** (FN);
- **ошибка II рода:** модель сказала «да» ($\hat{y} = 1$), хотя на самом деле «нет» ($y = 0$), т.е. **false positive** (FP).

Важно: если поменять классы местами (перекодировать метки), то ошибка I рода становится ошибкой II рода и наоборот.

Вероятности ошибок (rates): обозначим n_1 — число объектов класса C_1 , n_0 — число объектов класса C_0 . В лекциях вводятся

$$fnr = \frac{FN}{TP + FN} = \frac{FN}{n_1} \quad (\text{ошибка I рода, false negative rate}),$$

$$fpr = \frac{FP}{TN + FP} = \frac{FP}{n_0} \quad (\text{ошибка II рода, false positive rate}).$$

Отсюда

$$1 - fnr = \frac{TP}{TP + FN}$$

называют **полнотой** (*recall*, иногда также *sensitivity*).

ROC–кривая (Receiver Operating Characteristic): характеризует классификатор «в целом» при изменении порога (threshold) t . Идея такая: если у модели есть метрика качества «насколько объект похож на C_1 », то при увеличении/уменьшении порога t меняется, сколько объектов мы относим к C_1 . При этом обычно растут и TP , и FP (не ошибается только тот, кто ничего не делает), поэтому одновременно растут $fpr(t)$ и $recall(t)$.

Формально ROC задаётся как параметрическая кривая

$$(fpr(t), recall(t)), \quad t \in \mathbb{R},$$

которая соединяет точки $(0, 0)$ и $(1, 1)$ на плоскости $(fpr, recall)$.

- Крайность 1: классификатор всегда выдаёт $\hat{y} \equiv 0$. Тогда $TP = FP = 0$, поэтому $(fpr, recall) = (0, 0)$.
- Крайность 2: классификатор всегда выдаёт $\hat{y} \equiv 1$. Тогда $TN = FN = 0$, поэтому $(fpr, recall) = (1, 1)$.

Все разумные настройки порога дают точки «между» этими крайностями.

Как выглядит ROC–кривая:

- **случайный классификатор** (подбрасываем «монетку» с вероятностью p отнести к C_1) даёт $fpr(p) = recall(p) = p$, то есть ROC — **диагональ** квадрата;

- «хороший» классификатор стремится давать кривую, которая лежит **выше диагонали** (больше recall при том же fpr);
- выбор *между алгоритмами/настройками* — это выбор ROC-кривой, а выбор *порога/threshold* — выбор конкретной точки на выбранной кривой.

Числовой показатель качества: стандартно используют **AUC** (area under curve) — площадь под ROC-кривой. Интуитивно: чем больше площадь (чем «выше» кривая над диагональю), тем лучше классификатор «в целом». Для ориентира: у диагонали (случайный классификатор) AUC близка к 0.5, а у почти идеального классификатора — близка к 1.

3 Что такое recall и precision? Как устроена PR-кривая и что она показывается? Что такое F1 и F_β -меры, зачем они нужны?

Recall (полнота, sensitivity): доля объектов положительного класса, которые модель нашла:

$$\text{recall} = \frac{TP}{TP + FN}.$$

Precision (точность/прецизионность): доля правильных среди всех объектов, которые модель объявила положительными:

$$\text{precision} = \frac{TP}{TP + FP}.$$

PR-кривая (precision-recall curve): как и ROC, строится при изменении порога t у скорингового классификатора. Для каждого t считаем точку

$$(\text{recall}(t), \text{precision}(t)).$$

Она показывает компромисс: обычно при росте recall падает precision (чем больше «ловим» положительных, тем больше среди них ложных тревог). В лекциях подчёркивается, что при сильном дисбалансе классов PR-кривая часто информативнее ROC.

F1-мера: гармоническое среднее precision и recall:

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Она нужна, когда важно одновременно и «не промахиваться» (precision), и «не пропускать» (recall).

F_β -мера: обобщение, которое задаёт, что важнее:

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}.$$

- $\beta > 1$ сильнее штрафует за низкий recall (важнее *не пропускать* положительный класс).
- $\beta < 1$ сильнее штрафует за низкий precision (важнее *не давать ложных тревог*).

4 Выведите формулу байесовского классификатора. Докажите, что он имеет наибольшую точность.

Обозначения (как в лекциях): $y \in \{0, 1\}$, классы C_0 и C_1 ; $\pi_k = \mathbb{P}(y = k)$ — априорные вероятности классов; $f_k(\vec{x}) = p(\vec{x} | y = k)$ — условные плотности.

Апостериорная вероятность (скор):

$$\mathbb{P}(y = 1 | \vec{x}) = \frac{\pi_1 f_1(\vec{x})}{\pi_0 f_0(\vec{x}) + \pi_1 f_1(\vec{x})} \equiv r(\vec{x}), \quad \mathbb{P}(y = 0 | \vec{x}) = 1 - r(\vec{x}).$$

Байесовский классификатор (для 0–1 потерь):

$$\hat{y}(\vec{x}) = \arg \max_{k \in \{0, 1\}} \mathbb{P}(y = k | \vec{x}) \iff \hat{y}(\vec{x}) = \mathbb{I}\{r(\vec{x}) > 1/2\}.$$

Эквивалентная форма через плотности (без знаменателя):

$$\hat{y}(\vec{x}) = 1 \iff \pi_1 f_1(\vec{x}) > \pi_0 f_0(\vec{x}).$$

(В многоклассовом случае: $\hat{y}(\vec{x}) = \arg \max_{k=1, \dots, K} \pi_k f_k(\vec{x})$.)

Почему он даёт наибольшую точность (краткое доказательство): рассмотрим 0–1 потери $\ell(\hat{y}, y) = \mathbb{I}\{\hat{y} \neq y\}$. Условный риск при фиксированном \vec{x} для решения $\hat{y} = k$ равен

$$R(k | \vec{x}) = \mathbb{E}[\ell(k, y) | \vec{x}] = \mathbb{P}(y \neq k | \vec{x}) = 1 - \mathbb{P}(y = k | \vec{x}).$$

Минимизировать $R(k | \vec{x})$ по k значит *максимизировать* $\mathbb{P}(y = k | \vec{x})$. Следовательно, байесовское правило минимизирует ожидаемую ошибку $\mathbb{E} \mathbb{I}\{\hat{y}(\vec{x}) \neq y\}$, а значит максимизирует accuracy (так как $\text{Acc} = 1 - (\text{доля ошибок})$).

5 Опишите наивный байесовский подход. Опишите классификаторы LDA и QDA. Выведите формулы LDA и QDA методом наибольшего правдоподобия.

Общая байесовская схема: при априорах π_k и плотностях $f_k(\vec{x}) = p(\vec{x} | y = k)$

$$\hat{y}(\vec{x}) = \arg \max_k \pi_k f_k(\vec{x}).$$

Наивный Байес: предположение условной независимости признаков при фиксированном классе:

$$\vec{x} = (x_1, \dots, x_m), \quad p(\vec{x} | y = k) = \prod_{j=1}^m p(x_j | y = k).$$

Отсюда правило (в лог-форме):

$$\hat{y}(\vec{x}) = \arg \max_k \left(\log \pi_k + \sum_{j=1}^m \log p(x_j | y = k) \right).$$

LDA/QDA: гауссовские классы. Предположение: $\vec{x} \mid (y = k) \sim \mathcal{N}(\vec{\mu}_k, \Sigma_k)$. Тогда дискриминант (лог-постериор с точностью до константы)

$$\delta_k(\vec{x}) = \log \pi_k - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (\vec{x} - \vec{\mu}_k)^\top \Sigma_k^{-1} (\vec{x} - \vec{\mu}_k), \quad \hat{y}(\vec{x}) = \arg \max_k \delta_k(\vec{x}).$$

QDA: ковариации разные (Σ_k), границы решений квадратичны.

LDA: общая ковариация ($\Sigma_k \equiv \Sigma$), тогда

$$\delta_k(\vec{x}) = \vec{x}^\top \Sigma^{-1} \vec{\mu}_k - \frac{1}{2} \vec{\mu}_k^\top \Sigma^{-1} \vec{\mu}_k + \log \pi_k$$

(линейно по \vec{x}).

Оценки МНП (MLE) по выборке $\{(\vec{x}_i, y_i)\}_{i=1}^n$: $n_k = \#\{i : y_i = k\}$.

$$\hat{\pi}_k = \frac{n_k}{n}, \quad \hat{\vec{\mu}}_k = \frac{1}{n_k} \sum_{i:y_i=k} \vec{x}_i.$$

QDA:

$$\hat{\Sigma}_k = \frac{1}{n_k} \sum_{i:y_i=k} (\vec{x}_i - \hat{\vec{\mu}}_k)(\vec{x}_i - \hat{\vec{\mu}}_k)^\top.$$

LDA (объединённая):

$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^K \sum_{i:y_i=k} (\vec{x}_i - \hat{\vec{\mu}}_k)(\vec{x}_i - \hat{\vec{\mu}}_k)^\top.$$

6 Опишите логистический классификатор. Дайте определение KL-дивергенции. Выведите с помощью нее функцию потерь классификатора.

Логистический классификатор (логистическая регрессия): строим линейную метрику качества

$$a(\vec{x}) = \theta_0 + \theta^\top \vec{x},$$

и интерпретируем его как вероятность положительного класса через сигмоиду

$$\hat{y}(\vec{x}) = \mathbb{P}(y = 1 \mid \vec{x}) = \sigma(a) = \frac{1}{1 + e^{-a}}.$$

Классификация по порогу $1/2$: $\hat{y}_{\text{class}}(\vec{x}) = \mathbb{I}\{\hat{y}(\vec{x}) > 1/2\} \iff a(\vec{x}) > 0$, граница решений — гиперплоскость $a(\vec{x}) = 0$.

KL-дивергенция (Кульбак–Лейблер): для истинного распределения P и приближения Q

$$\text{KL}(P\|Q) = \int p(z) \ln \frac{p(z)}{q(z)} dz \quad (\text{или } \sum_i p_i \ln \frac{p_i}{q_i} \text{ в дискретном случае}).$$

Вывод функции потерь через KL (как в лекциях): считаем P заданным нашими метками. Для каждого объекта x_i введём

$$P_i : p(Y = 1 \mid x_i) = y_i, \quad p(Y = 0 \mid x_i) = 1 - y_i \quad (y_i \in \{0, 1\}).$$

Модельное распределение (классификатор) задаём ответом $\hat{y}_i = \hat{y}(x_i)$:

$$Q_i : q(Y = 1 | x_i) = \hat{y}_i, \quad q(Y = 0 | x_i) = 1 - \hat{y}_i.$$

Так как в $\text{KL}(P\|Q) = \int \ln dP dP - \int \ln dQ dP$ первое слагаемое от Q не зависит, минимизируем

$$-\sum_{i=1}^n \left(y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i) \right).$$

Итого (усреднённая) функция потерь:

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^n \left(y_i \ln \hat{y}_i(\theta) + (1 - y_i) \ln(1 - \hat{y}_i(\theta)) \right).$$

Эквивалентная запись при кодировке $y_i \in \{-1, 1\}$:

$$\ell(\theta) = \frac{1}{n} \sum_{i=1}^n \ln(1 + e^{-y_i a_i}), \quad a_i = a(x_i).$$

7 Опишите алгоритм градиентного спуска на примере логистического классификатора. Проверьте функцию потерь на выпуклость.

Градиентный спуск (batch GD): для функции потерь $\ell(\theta)$ строим итерации

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \ell(\theta^{(t)}), \quad t = 0, 1, 2, \dots$$

где $\eta > 0$ — шаг (learning rate). Критерий остановки (как обычно в лекциях): малость $\|\nabla \ell(\theta^{(t)})\|$ и/или малость $\|\theta^{(t+1)} - \theta^{(t)}\|$.

Пример: логистическая регрессия. Пусть $\tilde{x}_i = (1, \vec{x}_i)$ (добавили константу под свободный член), $a_i = \theta^\top \tilde{x}_i$, $p_i = \sigma(a_i)$. Функция потерь (кросс-энтропия / отрицательное лог-правдоподобие):

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^n \left(y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \right).$$

Тогда градиент имеет вид

$$\nabla \ell(\theta) = \frac{1}{n} \sum_{i=1}^n (p_i - y_i) \tilde{x}_i.$$

Итерация GD:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{1}{n} \sum_{i=1}^n (p_i^{(t)} - y_i) \tilde{x}_i, \quad p_i^{(t)} = \sigma((\theta^{(t)})^\top \tilde{x}_i).$$

(Для SGD в лекциях: заменить сумму по всем i на один объект или мини-батч.)

Проверка выпуклости: выпишем гессиан. Обозначим матрицу признаков $\tilde{X} \in \mathbb{R}^{n \times (m+1)}$ (строки \tilde{x}_i^\top), и веса $w_i = p_i(1 - p_i) \geq 0$, $W = \text{diag}(w_1, \dots, w_n)$. Тогда

$$\nabla^2 \ell(\theta) = \frac{1}{n} \tilde{X}^\top W \tilde{X}.$$

Для любого вектора v имеем

$$v^\top \nabla^2 \ell(\theta) v = \frac{1}{n} \sum_{i=1}^n w_i (\tilde{x}_i^\top v)^2 \geq 0,$$

значит $\nabla^2 \ell(\theta) \succeq 0$ и функция $\ell(\theta)$ выпуклая.

8 Опишите задачу многоклассовой классификации и метод softmax regression. Выведите функцию потерь из метода наибольшего правдоподобия.

Многоклассовая классификация: $y \in \{1, \dots, K\}$; по признакам $\vec{x} \in \mathbb{R}^m$ строим модель вероятностей $\mathbb{P}(y = k \mid \vec{x})$ и предсказываем класс по правилу

$$\hat{y}(\vec{x}) = \arg \max_{k=1, \dots, K} \mathbb{P}(y = k \mid \vec{x}).$$

Softmax regression (многоклассовая логистическая регрессия): вводим линейные «логиты»

$$a_k(\vec{x}) = b_k + w_k^\top \vec{x}, \quad k = 1, \dots, K,$$

и задаём вероятности через softmax:

$$p_k(\vec{x}) = \mathbb{P}(y = k \mid \vec{x}) = \frac{e^{a_k(\vec{x})}}{\sum_{j=1}^K e^{a_j(\vec{x})}}.$$

Функция потерь из МНП (MLE): используем one-hot кодировку метки $y_{ik} = \mathbb{I}\{y_i = k\}$. Тогда правдоподобие

$$L(\Theta) = \prod_{i=1}^n \prod_{k=1}^K p_k(\vec{x}_i)^{y_{ik}},$$

лог-правдоподобие

$$\log L(\Theta) = \sum_{i=1}^n \sum_{k=1}^K y_{ik} \log p_k(\vec{x}_i),$$

и отрицательное среднее лог-правдоподобие (кросс-энтропия):

$$\ell(\Theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K y_{ik} \log p_k(\vec{x}_i).$$

Для одного объекта с истинным классом y_i это просто $\ell_i(\Theta) = -\log p_{y_i}(\vec{x}_i)$.

9 Поставьте задачу кластеризации, приведите примеры разных метрик. Дайте определения метрик качества: внутри- и межкластерное расстояние, WSS, BSS, Silhouette.

Задача кластеризации: по выборке $X = \{\vec{x}_i\}_{i=1}^n \subset \mathbb{R}^m$ (меток нет) разбить объекты на K кластеров C_1, \dots, C_K так, чтобы внутри кластера объекты были «похожи», а между кластерами — «непохожи». Обычно это формализуют через минимизацию внутрикластерного разброса (например, суммы квадратов расстояний до центра кластера).

Примеры метрик/мер близости $d(\cdot, \cdot)$:

- евклидова: $\|\vec{x} - \vec{z}\|_2$;
- манхэттенская: $\|\vec{x} - \vec{z}\|_1$;
- Минковского: $\|\vec{x} - \vec{z}\|_p$;
- Чебышёва: $\|\vec{x} - \vec{z}\|_\infty$;
- косинусная «дистанция»: $1 - \frac{\langle x, z \rangle}{\|x\| \|z\|}$;
- Махalanобиса: $\sqrt{(x - z)^\top S^{-1}(x - z)}$.

Внутрикластерное расстояние (within): характерный «разброс» внутри кластера. В лекциях часто берут через центр (прототип) $\vec{\mu}_k$:

$$D_{\text{within}}(C_k) = \sum_{i \in C_k} d(\vec{x}_i, \vec{\mu}_k) \quad \text{или} \quad \sum_{i \in C_k} \|\vec{x}_i - \vec{\mu}_k\|^2.$$

Межкластерное расстояние (between): мера «разделённости» кластеров, например

$$D_{\text{between}}(C_k, C_\ell) = d(\vec{\mu}_k, \vec{\mu}_\ell)$$

(или в иерархической кластеризации: min / max / avg попарных расстояний между точками кластеров).

WSS (within sum of squares): при евклидовой метрике и центрах $\vec{\mu}_k$ (обычно средних) определяется как

$$\text{WSS} = \sum_{k=1}^K \sum_{i \in C_k} \|\vec{x}_i - \vec{\mu}_k\|^2.$$

BSS (between sum of squares): пусть $\bar{\vec{x}} = \frac{1}{n} \sum_{i=1}^n \vec{x}_i$ — общий центр, $n_k = |C_k|$. Тогда

$$\text{BSS} = \sum_{k=1}^K n_k \|\vec{\mu}_k - \bar{\vec{x}}\|^2.$$

(В классической евклидовой постановке: TSS = WSS + BSS.)

Silhouette (силуэт): для объекта i : $a(i)$ — средняя дистанция до точек своего кластера, $b(i)$ — минимальная по другим кластерам средняя дистанция. Тогда

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, 1].$$

Суммарный индекс — среднее $\frac{1}{n} \sum_i s(i)$ (больше — лучше).

10 Опишите методы k -means и k -medoids. Приведите алгоритмы решения Lloyd и Elkan. Оцените их скорость.

k -means (евклидовая постановка): дано K . Ищем разбиение $\{C_k\}_{k=1}^K$ и центры $\{\vec{\mu}_k\}$, минимизируя

$$\text{WSS}(C, \mu) = \sum_{k=1}^K \sum_{i \in C_k} \|\vec{x}_i - \vec{\mu}_k\|^2.$$

Алгоритм Lloyd (стандартный k -means): повторять до сходимости:

1. *Assignment*: $C_k := \{i : k = \arg \min_j \|\vec{x}_i - \vec{\mu}_j\|^2\}$.
2. *Update*: $\vec{\mu}_k := \frac{1}{|C_k|} \sum_{i \in C_k} \vec{x}_i$.

Каждая итерация не увеличивает WSS, сходимость к локальному минимуму.

Скорость Lloyd: на итерации считаем расстояния до всех центров: $O(nKm)$ (в лекциях обычно d вместо m). Итого за T итераций: $O(T nKm)$.

Elkan (ускорение k -means): использует неравенство треугольника и хранит верхнюю границу u_i на расстояние до текущего центра и нижние границы l_{ij} до остальных центров. Если для некоторого j выполнено $u_i \leq l_{ij}$ (или эквивалентные условия через межцентровые расстояния), то расстояние $\|\vec{x}_i - \vec{\mu}_j\|$ можно не считать.

Скорость Elkan: в худшем случае асимптотика та же $O(T nKm)$, но на реальных данных обычно существенно меньше вычислений расстояний.

k -medoids: аналог k -means для произвольной метрики d . Вместо центров берём **медоиды** $m_k \in \{\vec{x}_i\}$ (обязаны быть точками выборки) и минимизируем

$$\sum_{k=1}^K \sum_{i \in C_k} d(\vec{x}_i, m_k).$$

Шаги похожи:

- *Assignment*: отнести x_i к ближайшему медоиду.
- *Update*: в каждом кластере выбрать медоид m_k как точку, минимизирующую сумму дистанций до остальных точек кластера.

По лекциям: k -medoids устойчивее к выбросам, но обновление дороже (обычно требуется перебор кандидатов внутри кластера).

11 Выведите EM–algorithm кластеризации для смеси нормальных распределений.

Модель GMM (смесь нормальных): вводим скрытую метку кластера $z_i \in \{1, \dots, K\}$ и параметры $\Theta = \{\pi_k, \vec{\mu}_k, \Sigma_k\}_{k=1}^K$, где $\pi_k \geq 0$, $\sum_k \pi_k = 1$. Предположение:

$$\mathbb{P}(z_i = k) = \pi_k, \quad \vec{x}_i | (z_i = k) \sim \mathcal{N}(\vec{\mu}_k, \Sigma_k).$$

Маргинальная плотность:

$$p(\vec{x}_i | \Theta) = \sum_{k=1}^K \pi_k \mathcal{N}(\vec{x}_i | \vec{\mu}_k, \Sigma_k), \quad \ell(\Theta) = \sum_{i=1}^n \log p(\vec{x}_i | \Theta).$$

EM–алгоритм: итеративно максимизируем лог-правдоподобие, чередуя:
E-step (responsibilities):

$$\gamma_{ik} := \mathbb{P}(z_i = k | \vec{x}_i, \Theta^{(t)}) = \frac{\pi_k^{(t)} \mathcal{N}(\vec{x}_i | \vec{\mu}_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^K \pi_j^{(t)} \mathcal{N}(\vec{x}_i | \vec{\mu}_j^{(t)}, \Sigma_j^{(t)})}.$$

Обозначим $N_k := \sum_{i=1}^n \gamma_{ik}$.

M-step (обновление параметров):

$$\pi_k^{(t+1)} = \frac{N_k}{n}, \quad \vec{\mu}_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^n \gamma_{ik} \vec{x}_i,$$

$$\Sigma_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^n \gamma_{ik} (\vec{x}_i - \vec{\mu}_k^{(t+1)}) (\vec{x}_i - \vec{\mu}_k^{(t+1)})^\top.$$

Свойство (по лекциям): на каждой итерации EM не уменьшает $\ell(\Theta)$ и сходится к стационарной точке (обычно локальному максимуму).

12 Опишите иерархические методы: ближнего соседа, дальнего соседа, средней связи, центроидов, Уорда. Выпишите формулу Ланса–Вильямса.

Иерархическая (агломеративная) кластеризация: стартуем с n кластеров–одиночек; на каждом шаге объединяем два «ближайших» кластера по правилу linkage, обновляем расстояния до нового кластера и продолжаем до одного кластера. Результат визуализируется *дендрограммой*; число кластеров выбирают «разрезом» дендрограммы.

Пусть $d(A, B)$ — расстояние между кластерами A и B (определяется правилом связи). Основные варианты (как в лекциях):

- **Ближний сосед (single linkage):** $d(A, B) = \min_{x \in A, z \in B} d(x, z)$.
- **Дальний сосед (complete linkage):** $d(A, B) = \max_{x \in A, z \in B} d(x, z)$.

- **Средняя связь (average linkage):** $d(A, B) = \frac{1}{|A||B|} \sum_{x \in A} \sum_{z \in B} d(x, z)$.
- **Центроиды (centroid linkage):** $d(A, B) = \|\mu_A - \mu_B\|$ (обычно евклидова), где μ_A — центр (среднее) кластера.
- **Уорд (Ward):** объединяя пару, дающую минимальный прирост внутрикластерной суммы квадратов (WSS); эквивалентно используют специальную «дистанцию Уорда».

Формула Ланса–Вильямса: пусть на шаге объединили кластеры A и B в $C = A \cup B$. Тогда расстояние до любого другого кластера S обновляется по схеме

$$d(C, S) = \alpha_A d(A, S) + \alpha_B d(B, S) + \beta d(A, B) + \gamma |d(A, S) - d(B, S)|.$$

Коэффициенты зависят от linkage. В обозначениях $n_A = |A|$, $n_B = |B|$, $n_C = n_A + n_B$:

- **Single:** $\alpha_A = \alpha_B = \frac{1}{2}$, $\beta = 0$, $\gamma = -\frac{1}{2}$.
- **Complete:** $\alpha_A = \alpha_B = \frac{1}{2}$, $\beta = 0$, $\gamma = +\frac{1}{2}$.
- **Average (UPGMA):** $\alpha_A = \frac{n_A}{n_C}$, $\alpha_B = \frac{n_B}{n_C}$, $\beta = 0$, $\gamma = 0$.
- **Centroid:** $\alpha_A = \frac{n_A}{n_C}$, $\alpha_B = \frac{n_B}{n_C}$, $\beta = -\frac{n_A n_B}{n_C^2}$, $\gamma = 0$.
- **Ward:** $\alpha_A = \frac{n_A + n_S}{n_C + n_S}$, $\alpha_B = \frac{n_B + n_S}{n_C + n_S}$, $\beta = -\frac{n_S}{n_C + n_S}$, $\gamma = 0$.

13 Опишите алгоритм DBScan. Опишите алгоритм OPTICS.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): задаём два параметра: радиус ε и MinPts. Определим ε -окрестность точки

$$N_\varepsilon(x) = \{z : d(x, z) \leq \varepsilon\}.$$

- **Core point:** $|N_\varepsilon(x)| \geq \text{MinPts}$.
- **Border point:** не core, но лежит в ε -окрестности некоторой core-точки.
- **Noise (выброс):** не core и не border.

Идея: кластер — это связная компонента по отношению «достижимости по плотности».

Алгоритм (как в лекциях):

1. Все точки пометить как непосещённые.
2. Для каждой непосещённой точки x :
 - (a) пометить x как посещённую, посчитать $N_\varepsilon(x)$;
 - (b) если $|N_\varepsilon(x)| < \text{MinPts}$, пометить x как noise (временно);

- (c) иначе создать новый кластер и *расширять* его: добавлять соседей, и для каждой найденной core-точки добавлять её соседей (BFS/очередь).

Плюсы (по лекциям): находит кластеры произвольной формы и автоматически выделяет шум.

OPTICS (Ordering Points To Identify the Clustering Structure): обобщение DBSCAN: строит *упорядочивание* точек и «профиль плотности», позволяя извлекать кластеры при разных ε . Параметры: MinPts и ε_{\max} (верхняя граница радиуса поиска).

Для точки x :

- **core-distance:** $\text{core_dist}(x)$ — расстояние до MinPts-го соседа (если core; иначе ∞).
- **reachability-distance (из x в z):**

$$\text{reach_dist}(x, z) = \max\{\text{core_dist}(x), d(x, z)\}.$$

Алгоритм: обход точек как в DBSCAN, но при расширении ведём приоритетную очередь по минимальной достижимости: для соседей z обновляем их текущую reach_dist и извлекаем следующую точку с минимальным значением. Выход: порядок точек + значения *reachability* (*reachability plot*); кластеры читаются как «впадины» на графике.

14 Сформулируйте алгоритм спектральной кластеризации. Выведите его на примере двух классов.

Идея спектральной кластеризации: переводим данные в граф: вершины — объекты, ребро отражает похожесть. Строим матрицу весов $W = (w_{ij})$ (например, по k NN или гауссовому ядру) и степени вершин $d_i = \sum_j w_{ij}$, $D = \text{diag}(d_1, \dots, d_n)$.

Лапласиан графа:

$$L = D - W \quad (\text{ненормированный}), \quad L_{\text{sym}} = I - D^{-1/2} W D^{-1/2} \quad (\text{симметрично-нормированный}).$$

Алгоритм (Ng–Jordan–Weiss / нормированный вариант, как в лекциях):

1. Построить W , затем D и L_{sym} .
2. Найти K собственных векторов u_1, \dots, u_K матрицы L_{sym} , соответствующих наименьшим собственным значениям.
3. Собрать матрицу $U \in \mathbb{R}^{n \times K}$ из строк $U_{\cdot i} = (u_1(i), \dots, u_K(i))$.
4. Нормировать строки: $y_i = \frac{U_{\cdot i}}{\|U_{\cdot i}\|}$.
5. Запустить k -means по точкам $\{y_i\}_{i=1}^n$ и получить кластеры в исходных данных.

Вывод для двух классов ($K = 2$): в нормированной постановке минимизация разреза (Ncut) после релаксации сводится к задаче на собственные векторы лапласиана. Для $K = 2$ берём второй по величине «малости» собственный вектор (*вектор Фидлера*) u_2 и делим вершины по его знаку:

$$C_1 = \{i : u_2(i) \geq 0\}, \quad C_2 = \{i : u_2(i) < 0\}$$

(или по порогу, если в лекциях используется не нулевой threshold).

15 Опишите алгоритм понижения размерности. Выведите формулу для РСА.

Понижение размерности: по данным $\vec{x} \in \mathbb{R}^m$ построить отображение $\Phi : \mathbb{R}^m \rightarrow \mathbb{R}^r$ (обычно $r \ll m$), сохраняющее «важную» структуру данных (вариативность/расстояния/кластеры) и уменьшающее размерность.

РСА (метод главных компонент): ищем r -мерное линейное подпространство, на которое проекция данных имеет максимальную дисперсию (эквивалентно — минимизирует среднеквадратичную ошибку восстановления).

Вывод (как в лекциях): центруем данные: $\bar{\vec{x}} = \frac{1}{n} \sum_{i=1}^n \vec{x}_i$, $\tilde{\vec{x}}_i = \vec{x}_i - \bar{\vec{x}}$. Соберём матрицу $\tilde{X} \in \mathbb{R}^{n \times m}$ из строк $\tilde{\vec{x}}_i^\top$. Ковариационная матрица (с точностью до константы):

$$S = \frac{1}{n} \tilde{X}^\top \tilde{X}.$$

Ищем направления v единичной длины, максимизирующие дисперсию проекций:

$$\max_{\|v\|=1} \text{Var}(\tilde{X}v) = \max_{\|v\|=1} v^\top S v.$$

По методу множителей Лагранжа получаем $Sv = \lambda v$, т.е. v — собственный вектор S .

Формула РСА: пусть $\lambda_1 \geq \dots \geq \lambda_m$ — собственные значения S , $V_r = [v_1, \dots, v_r]$ — матрица из r собственных векторов, соответствующих наибольшим λ . Тогда проекция (главные компоненты)

$$Z = \tilde{X}V_r \in \mathbb{R}^{n \times r},$$

а восстановление из r компонент:

$$\hat{\vec{x}} = ZV_r^\top + \mathbf{1}\bar{\vec{x}}^\top.$$

(Эквивалентно через SVD: $\tilde{X} = U\Sigma V^\top$, берём первые r столбцов V .)

16 Выведите формулу для вероятностного РСА. Как выбирать число измерений проекции?

Вероятностный РСА (PPCA): вводим скрытую переменную $z \in \mathbb{R}^r$ и линейную генеративную модель

$$\vec{x} = W\vec{z} + \vec{\mu} + \vec{\varepsilon}, \quad \vec{z} \sim \mathcal{N}(0, I_r), \quad \vec{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I_m).$$

Тогда маргинальное распределение наблюдений нормальное:

$$\vec{x} \sim \mathcal{N}(\vec{\mu}, C), \quad C = WW^\top + \sigma^2 I_m.$$

Оценки МНП (MLE) (как в лекциях): пусть S — ковариация центрированных данных, $S = U\Lambda U^\top$, $\Lambda = \text{diag}(\lambda_1 \geq \dots \geq \lambda_m)$. Тогда

$$\hat{\sigma}^2 = \frac{1}{m-r} \sum_{j=r+1}^m \lambda_j,$$

а матрица нагрузок (с точностью до вращения $R \in \mathbb{R}^{r \times r}$, $R^\top R = I$)

$$\hat{W} = U_r (\Lambda_r - \hat{\sigma}^2 I_r)^{1/2} R,$$

где U_r — первые r собственных векторов, $\Lambda_r = \text{diag}(\lambda_1, \dots, \lambda_r)$.

Проекция (постериорное среднее): для центрированного $\tilde{x} = \vec{x} - \vec{\mu}$

$$\mathbb{E}[\vec{z} | \vec{x}] = M^{-1} \hat{W}^\top \tilde{x}, \quad M = \hat{W}^\top \hat{W} + \hat{\sigma}^2 I_r.$$

Как выбирать r (размерность проекции):

- по доле объяснённой дисперсии (как в PCA): $\frac{\sum_{j=1}^r \lambda_j}{\sum_{j=1}^m \lambda_j} \geq \tau$ (например, $\tau = 0.9$);
- по «локтю» (scree plot) по спектру λ_j ;
- по качеству восстановления / CV (ошибка реконструкции vs r);
- для RPCA можно также сравнивать модели по лог-правдоподобию с штрафом за сложность (AIC/BIC).

17 Опишите нелинейный алгоритм PCA. Приведите примеры разных ядер.

Нелинейный PCA = Kernel PCA: заменяем явное отображение $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$ (в большое/бесконечное пространство признаков) на ядро $k(x, z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{H}}$. Далее делаем обычный PCA, но в пространстве \mathcal{H} .

Алгоритм Kernel PCA (по лекциям):

1. По данным x_1, \dots, x_n построить матрицу Грама $K \in \mathbb{R}^{n \times n}$: $K_{ij} = k(x_i, x_j)$.
2. Отцентрировать в \mathcal{H} (центрирование ядра):

$$K_c = K - \mathbf{1}K - K\mathbf{1} + \mathbf{1}\mathbf{1}^\top, \quad \mathbf{1} := \frac{1}{n}\mathbf{e}\mathbf{e}^\top.$$

3. Решить спектральную задачу

$$K_c \alpha = n\lambda \alpha,$$

взять r наибольших λ и соответствующие векторы $\alpha^{(1)}, \dots, \alpha^{(r)}$.

4. Нормировка (как обычно): $\alpha^{(k)} \leftarrow \alpha^{(k)} / \sqrt{n\lambda_k}$.
5. Координаты (проекции) обучающей точки x_i на k -ю компоненту: $z_{ik} = \alpha_i^{(k)} n\lambda_k$ (эквивалентно: $z^{(k)} = K_c \alpha^{(k)}$).

Проекция нового объекта x : считаем вектор ядерных значений $k_x = (k(x_1, x), \dots, k(x_n, x))^\top$, центрируем его так же, как обучающее K (через средние по строкам/столбцам), и берём

$$z_k(x) = \sum_{i=1}^n \alpha_i^{(k)} k_c(x_i, x).$$

Примеры ядер:

- линейное: $k(x, z) = x^\top z$ (даёт обычный РСА);
- полиномиальное: $k(x, z) = (x^\top z + c)^p$;
- гауссово (RBF): $k(x, z) = \exp(-\|x - z\|^2/(2\sigma^2))$;
- сигмоидальное: $k(x, z) = \tanh(\kappa x^\top z + \theta)$;
- лапласовское: $k(x, z) = \exp(-\|x - z\|_1/\sigma)$.

18 Дайте постановку метода SVC с мягким и жестким зазором. Сведите обе задачи к задачам условной оптимизации.

SVC (SVM) для бинарной классификации: пусть $y_i \in \{-1, +1\}$, $f(x) = w^\top x + b$. Классификатор: $\hat{y}(x) = \text{sign}(f(x))$.

Жёсткий зазор (hard margin): требуем линейную разделимость и максимизируем зазор $2/\|w\|$. Эквивалентная задача условной оптимизации (primal):

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^\top x_i + b) \geq 1, \quad i = 1, \dots, n.$$

Мягкий зазор (soft margin): вводим послабления $\xi_i \geq 0$ (нарушение ограничений) и штраф $C > 0$. Условная оптимизация:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n. \end{aligned}$$

Эквивалентная безусловная форма (как в лекциях через hinge-loss): из ограничений следует $\xi_i = \max\{0, 1 - y_i f(x_i)\}$, поэтому

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i(w^\top x_i + b)).$$

19 Сформулируйте теорему Куна–Таккера. Сведите задачу SVC с жестким зазором к двойственной задаче.

Теорема Куна–Таккера (ККТ) (в лекционном виде): рассмотрим задачу

$$\min_x f(x) \quad \text{s.t.} \quad g_i(x) \leq 0 \quad (i = 1..m), \quad h_j(x) = 0 \quad (j = 1..p).$$

Лагранжиан: $\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \nu_j h_j(x)$. Если f и g_i выпуклы, h_j аффинны и выполнено условие Слейтера, то x^* оптимально тогда и только тогда, когда существуют множители $\lambda^* \geq 0, \nu^*$ такие, что выполнены условия ККТ:

- (primal feasible) $g_i(x^*) \leq 0, h_j(x^*) = 0$;
- (dual feasible) $\lambda_i^* \geq 0$;
- (stationarity) $\nabla_x \mathcal{L}(x^*, \lambda^*, \nu^*) = 0$;
- (complementary slackness) $\lambda_i^* g_i(x^*) = 0$ для всех i .

Двойственная задача для hard-margin SVC: primal (из предыдущего пункта)

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad 1 - y_i(w^\top x_i + b) \leq 0, \quad i = 1, \dots, n.$$

Лагранжиан при множителях $\alpha_i \geq 0$:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 + \sum_{i=1}^n \alpha_i (1 - y_i(w^\top x_i + b)).$$

Условия стационарности:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i, \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0.$$

Подставляя w в \mathcal{L} , получаем двойственную задачу:

$$\begin{aligned} \max_{\alpha} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{s.t.} & \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0. \end{aligned}$$

Комплементарная нежёсткость: $\alpha_i (y_i(w^\top x_i + b) - 1) = 0$, поэтому $\alpha_i > 0$ только у опорных векторов. Классификатор: $f(x) = \sum_i \alpha_i y_i \langle x_i, x \rangle + b$.

20 Опишите и выведите нелинейный метод SVC для задачи с жестким зазором.

Нелинейный SVC (hard margin) = ядерный SVM: вместо линейного разделения в \mathbb{R}^m переходим к отображению $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$ в (возможно бесконечномерное) пространство признаков. Решение ищем гиперплоскостью в \mathcal{H} : $f(x) = \langle w, \phi(x) \rangle_{\mathcal{H}} + b$, $\hat{y}(x) = \text{sign}(f(x))$.

Primal (условная оптимизация в \mathcal{H}):

$$\min_{w,b} \frac{1}{2} \|w\|_{\mathcal{H}}^2 \quad \text{s.t.} \quad y_i (\langle w, \phi(x_i) \rangle_{\mathcal{H}} + b) \geq 1, \quad i = 1, \dots, n.$$

Лагранжиан: при множителях $\alpha_i \geq 0$

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|_{\mathcal{H}}^2 + \sum_{i=1}^n \alpha_i (1 - y_i (\langle w, \phi(x_i) \rangle_{\mathcal{H}} + b)).$$

Условия стационарности:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i \phi(x_i), \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0.$$

Двойственная задача (kernel trick): вводим ядро $k(x, z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{H}}$. Представляя w в лагранжиан, получаем

$$\begin{aligned} \max_{\alpha} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{s.t. } & \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0. \end{aligned}$$

Решающее правило:

$$f(x) = \sum_{i=1}^n \alpha_i y_i k(x_i, x) + b.$$

По ККТ: $\alpha_i > 0$ только у опорных векторов.

21 Сформулируйте теорему Мерсера. Приведите примеры ядер для нелинейного SVC.

Теорема Мерсера (в «прикладной» формулировке из лекций): ядро $k(x, z)$ допустимо для kernel-методов (т.е. существует отображение ϕ такое, что $k(x, z) = \langle \phi(x), \phi(z) \rangle$) тогда и только тогда, когда оно симметрично $k(x, z) = k(z, x)$ и положительно полуопределен: для любых точек x_1, \dots, x_n матрица Грама

$$K_{ij} = k(x_i, x_j)$$

удовлетворяет $K \succeq 0$ (то есть $\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0$ для любых $c \in \mathbb{R}^n$).

Эквивалентная «интегральная» версия: если k непрерывно и симметрично на компактном множестве, то существует разложение

$$k(x, z) = \sum_{t=1}^{\infty} \lambda_t \psi_t(x) \psi_t(z), \quad \lambda_t \geq 0,$$

где $\{\psi_t\}$ — ортонормированные собственные функции соответствующего интегрального оператора.

Примеры ядер для нелинейного SVC:

- **линейное:** $k(x, z) = x^\top z$;
- **полиномиальное:** $k(x, z) = (x^\top z + c)^p$, $p \in \mathbb{N}$, $c \geq 0$;
- **гауссово (RBF):** $k(x, z) = \exp(-\|x - z\|^2/(2\sigma^2))$;
- **лапласовское:** $k(x, z) = \exp(-\|x - z\|_1/\sigma)$;
- **сигмоидальное:** $k(x, z) = \tanh(\kappa x^\top z + \theta)$ (используют, но корректность как Mercer-ядра зависит от параметров).

22 Сведите задачу многоклассовой классификации к бинарной методами one-versus-one и one-versus-all.

Постановка: многоклассовая классификация $y \in \{1, \dots, K\}$. Идея редукции: заменить одну K -классовую задачу набором бинарных и собрать решение по их ответам.

One-versus-all (OvA, one-vs-rest): строим K бинарных классификаторов $g_k(x)$, где положительный класс — C_k , отрицательный — «все остальные» $\cup_{j \neq k} C_j$. На тесте получаем оценки (метрика качества/маржу/вероятность) $s_k(x)$ и выбираем

$$\hat{y}(x) = \arg \max_{k=1, \dots, K} s_k(x).$$

(В лекциях: при калиброванных вероятностях можно брать $\arg \max_k \mathbb{P}(y = k \mid x)$.) Число моделей: K .

One-versus-one (OvO): строим бинарный классификатор для каждой пары классов (k, ℓ) : обучение ведём только на объектах из $C_k \cup C_\ell$. Всего моделей: $\binom{K}{2} = K(K - 1)/2$. На тесте каждый классификатор «голосует» за один из двух классов; итоговый класс выбираем по большинству голосов:

$$\hat{y}(x) = \arg \max_k \#\{\ell \neq k : g_{k\ell}(x) = k\}.$$

(В лекциях иногда упоминают вариант с суммой маржин/вероятностей вместо простого голосования.)

Сравнение (по лекциям, кратко): OvA — меньше моделей, но каждый классификатор видит «смешанный» отрицательный класс; OvO — больше моделей, но каждая бинарная задача проще (только две категории) и часто стабильнее для SVM.

23 Дайте постановку метода SVR. Сведите задачи с мягким и жестким зазором к задачам условной оптимизации.

SVR (Support Vector Regression): строим регрессионную функцию вида

$$f(x) = w^\top x + b$$

и хотим, чтобы предсказания лежали внутри “ ε -трубки” вокруг истинных значений y_i . Используется ε -нечувствительная ошибка: $|y - f(x)|_\varepsilon := \max\{0, |y - f(x)| - \varepsilon\}$.

Жёсткий зазор (hard margin SVR): требуем, чтобы все точки попадали в ε -трубку. *Условная оптимизация* (primal):

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad \begin{cases} y_i - (w^\top x_i + b) \leq \varepsilon, \\ (w^\top x_i + b) - y_i \leq \varepsilon, \end{cases} \quad i = 1, \dots, n.$$

Эквивалентно: $|y_i - f(x_i)| \leq \varepsilon$ для всех i .

Мягкий зазор (soft margin SVR): разрешаем выходить за ε -трубку, вводя послабления $\xi_i, \xi_i^* \geq 0$ и штраф $C > 0$. Условная оптимизация:

$$\begin{aligned} & \min_{w, b, \xi, \xi^*} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \\ \text{s.t. } & \begin{cases} y_i - (w^\top x_i + b) \leq \varepsilon + \xi_i, \\ (w^\top x_i + b) - y_i \leq \varepsilon + \xi_i^*, \\ \xi_i \geq 0, \quad \xi_i^* \geq 0, \end{cases} \quad i = 1, \dots, n. \end{aligned}$$

(В лекциях: это эквивалентно минимизации суммы ε -нечувствительных потерь с регуляризацией $\frac{1}{2} \|w\|^2$.)

24 Сформулируйте теорему Куна–Таккера. Выведите двойственную задачу для метода SVR с мягким зазором. Опишите ядерный метод SVR.

ККТ (напоминание): для выпуклой задачи при выполнении условия Слейтера оптимум описывается условиями: допустимость (primal/dual), стационарность $\nabla_x \mathcal{L} = 0$ и комплементарная нежёсткость.

SVR с мягким зазором (primal): (из предыдущего пункта)

$$\begin{aligned} & \min_{w, b, \xi, \xi^*} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \\ \text{s.t. } & \begin{cases} y_i - (w^\top x_i + b) \leq \varepsilon + \xi_i, \\ (w^\top x_i + b) - y_i \leq \varepsilon + \xi_i^*, \\ \xi_i \geq 0, \quad \xi_i^* \geq 0. \end{cases} \end{aligned}$$

Двойственная задача (вывод через Лагранжиан): вводим множители $\alpha_i, \alpha_i^* \geq 0$ для первых двух ограничений и $\eta_i, \eta_i^* \geq 0$ для неотрицательности ξ_i, ξ_i^* . Лагранжиан:

$$\mathcal{L} = \frac{1}{2} \|w\|^2 + C \sum_i (\xi_i + \xi_i^*) + \sum_i \alpha_i (y_i - \varepsilon - \xi_i - (w^\top x_i + b)) + \sum_i \alpha_i^* ((w^\top x_i + b) - y_i - \varepsilon - \xi_i^*) - \sum_i \eta_i \xi_i - \sum_i \eta_i^* \xi_i^*$$

Стационарность:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_i (\alpha_i - \alpha_i^*) x_i, \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_i (\alpha_i - \alpha_i^*) = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \eta_i = 0 \Rightarrow 0 \leq \alpha_i \leq C, \quad \frac{\partial \mathcal{L}}{\partial \xi_i^*} = 0 \Rightarrow C - \alpha_i^* - \eta_i^* = 0 \Rightarrow 0 \leq \alpha_i^* \leq C.$$

Подставляя w , получаем двойственную задачу:

$$\max_{\alpha, \alpha^*} -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle x_i, x_j \rangle - \varepsilon \sum_{i=1}^n (\alpha_i + \alpha_i^*) + \sum_{i=1}^n y_i (\alpha_i - \alpha_i^*)$$

$$\text{s.t. } \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad 0 \leq \alpha_i \leq C, \quad 0 \leq \alpha_i^* \leq C.$$

Решающее правило (линейный SVR):

$$f(x) = \sum_{i=1}^n (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b.$$

По ККТ ненулевые $(\alpha_i - \alpha_i^*)$ соответствуют *опорным векторам* (точкам на/вне ε -трубки). b находят из ККТ, используя любой объект с $0 < \alpha_i < C$ или $0 < \alpha_i^* < C$: $y_i - f(x_i) = \varepsilon$ или $f(x_i) - y_i = \varepsilon$.

Ядерный SVR: заменяем скалярное произведение на ядро $k(x, z) = \langle \phi(x), \phi(z) \rangle$. Тогда в двойственной задаче просто $\langle x_i, x_j \rangle \rightarrow k(x_i, x_j)$, а предсказание

$$f(x) = \sum_{i=1}^n (\alpha_i - \alpha_i^*) k(x_i, x) + b.$$

25 Опишите метод KNN. Приведите примеры различных метрик. Что такое весовой KNN?

kNN (k-nearest neighbors, метод k ближайших соседей): непараметрический метод метрического обучения: для нового объекта x ищем k ближайших к нему обучающих объектов по выбранной метрике $d(\cdot, \cdot)$ и агрегируем их ответы.

Алгоритм (по лекциям):

1. Выбрать k и метрику d .
2. Для запроса x посчитать расстояния $d(x, x_i)$ до всех объектов обучающей выборки.
3. Взять множество индексов $\mathcal{N}_k(x) — k$ ближайших.
4. **Классификация:** выбрать класс по большинству голосов:

$$\hat{y}(x) = \arg \max_c \sum_{i \in \mathcal{N}_k(x)} \mathbb{I}\{y_i = c\}.$$

Регрессия: усреднить ответы:

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i.$$

Примеры метрик/мер близости:

- евклидова: $\|x - z\|_2$;
- манхэттенская: $\|x - z\|_1$;
- Минковского: $\|x - z\|_p$;

- Чебышёва: $\|x - z\|_\infty$;
- косинусная «дистанция»: $1 - \frac{\langle x, z \rangle}{\|x\| \|z\|}$;
- Махalanобиса: $\sqrt{(x - z)^\top S^{-1}(x - z)}$.

Весовой kNN: соседи голосуют/усредняются с весами $w_i(x)$, зависящими от расстояния (ближе — больше вес).

- **Классификация:**

$$\hat{y}(x) = \arg \max_c \sum_{i \in \mathcal{N}_k(x)} w_i(x) \mathbb{I}\{y_i = c\}.$$

- **Регрессия:**

$$\hat{y}(x) = \frac{\sum_{i \in \mathcal{N}_k(x)} w_i(x) y_i}{\sum_{i \in \mathcal{N}_k(x)} w_i(x)}.$$

Типичные веса (как в лекциях): $w_i(x) = \frac{1}{d(x, x_i) + \delta}$ (малое $\delta > 0$ для устойчивости) или через ядро $w_i(x) = K(d(x, x_i)/h)$ (связь со «скользящим окном»).

Замечания: выбор k обычно делают по CV; наивная сложность запроса $O(nm)$ по вычислению расстояний (ускорение: k-d tree / ball tree / ANN).

26 Опишите метод скользящего окна для метрической классификации и регрессии. Приведите примеры различных ядер.

Метод скользящего окна (Parzen window / kernel method) в метрической постановке: задаём метрику $d(\cdot, \cdot)$ и ширину окна (bandwidth) $h > 0$. Для запроса x каждой обучающей точке x_i назначаем вес

$$w_i(x) = K\left(\frac{d(x, x_i)}{h}\right),$$

где $K(\cdot) \geq 0$ — ядро (обычно убывает с ростом аргумента). Интуиция: ближе x_i к x — больше вклад.

Классификация (скользящее окно): вес «за класс» равен сумме весов точек этого класса:

$$S_c(x) = \sum_{i=1}^n w_i(x) \mathbb{I}\{y_i = c\}, \quad \hat{y}(x) = \arg \max_c S_c(x).$$

(В лекциях: это эквивалентно plug-in байесовскому правилу через оценку условных плотностей Парзена по классам.)

Регрессия (Надарая—Уотсон):

$$\hat{y}(x) = \frac{\sum_{i=1}^n w_i(x) y_i}{\sum_{i=1}^n w_i(x)}.$$

Два варианта окна (как обычно в лекциях):

- **фиксированное окно:** h фиксировано;
- **переменное окно:** $h = h(x)$ берут как расстояние до k -го соседа (связь с k NN).

Примеры ядер $K(t)$:

- **прямоугольное (uniform window):** $K(t) = \mathbb{I}\{t \leq 1\}$;
- **треугольное:** $K(t) = \max\{0, 1 - t\}$;
- **Эпанечникова:** $K(t) = \max\{0, 1 - t^2\}$;
- **гауссово:** $K(t) = \exp(-t^2/2)$;
- **лапласовское:** $K(t) = \exp(-|t|)$.

Роль h : малое h — мало сглаживания (риск переобучения), большое h — сильное сглаживание (риск недообучения).

27 Дайте математическое (вероятностное) обоснование методов KNN и скользящего окна классификации и регрессии.

Идея (по лекциям): и k NN, и «скользящее окно» — это *локальные plug-in оценки* байесовских правил (через оценку постериоров/плотностей) и/или оценка регрессионной функции $m(x) = \mathbb{E}[Y | X = x]$.

Классификация: байесовское правило. Для K классов оптимальный по 0–1 потере классификатор:

$$\hat{y}(x) = \arg \max_{c \in \{1, \dots, K\}} \mathbb{P}(Y = c | X = x) = \arg \max_c \pi_c f_c(x),$$

где $\pi_c = \mathbb{P}(Y = c)$, $f_c(x) = p(x | Y = c)$. Обоснование k NN/окон: заменить неизвестные величины на локальные оценки.

Скользящее окно: оценка плотностей Парзена (KDE). Для каждого класса оцениваем условную плотность:

$$\hat{f}_c(x) = \frac{1}{n_c h^m} \sum_{i: y_i=c} K\left(\frac{x - x_i}{h}\right), \quad n_c = \#\{i : y_i = c\}, \quad h > 0.$$

Тогда plug-in классификатор:

$$\hat{y}(x) = \arg \max_c \hat{\pi}_c \hat{f}_c(x), \quad \hat{\pi}_c = \frac{n_c}{n}.$$

(Эквивалентно формуле из предыдущего пункта про окно: суммируем веса по классам.)

k NN: оценка плотности через объём шара. Пусть $r_k(x)$ — расстояние от x до k -го соседа (в выбранной метрике), V_m — объём единичного шара. Тогда локальная оценка плотности (интуиция « k точек в шаре радиуса r_k »):

$$\hat{f}(x) \approx \frac{k}{n V_m r_k(x)^m}, \quad \hat{f}_c(x) \approx \frac{k_c(x)}{n_c V_m r_k(x)^m},$$

где $k_c(x)$ — число соседей класса c среди k ближайших. Подстановка в $\arg \max_c \pi_c f_c(x)$ даёт

$$\hat{y}(x) = \arg \max_c k_c(x),$$

т.е. *голосование большинства* (весовой вариант получается, если вместо «шара» брать яdroвые веса).

Регрессия: условное математическое ожидание. Для квадратичной потери оптимальный прогноз (байесовский регрессор):

$$m(x) = \arg \min_a \mathbb{E}[(Y - a)^2 | X = x] = \mathbb{E}[Y | X = x].$$

Локальные методы оценивают $m(x)$ через локальное усреднение (закон больших чисел):

- **k NN-регрессия:** $\hat{m}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$;
- **окно (Надарая–Уотсон):** $\hat{m}(x) = \frac{\sum_i w_i(x) y_i}{\sum_i w_i(x)}$, $w_i(x) = K(d(x, x_i)/h)$.

Условия состоятельности (как обычно формулируют в лекциях): для сходимости к байесовскому решению берут $h \rightarrow 0$, $nh^m \rightarrow \infty$ (для окна) или $k \rightarrow \infty$, $k/n \rightarrow 0$ (для k NN).

28 Выведите формулы локальной регрессии из МНК. Сравните локально–постоянную и локально–линейную регрессии.

Локальная регрессия как взвешенный МНК (по лекциям): фиксируем точку запроса x и задаём веса $w_i(x) \geq 0$ (обычно через окно/ядро) $w_i(x) = K(d(x, x_i)/h)$. Далее подбираем параметры локальной модели, минимизируя *взвешенную* сумму квадратов.

Локально–постоянная регрессия (Nadaraya–Watson): берём модель $f(z) \equiv a$ в окрестности x и решаем

$$\hat{a}(x) = \arg \min_{a \in \mathbb{R}} \sum_{i=1}^n w_i(x) (y_i - a)^2.$$

Условие первого порядка даёт

$$\sum_i w_i(x) (y_i - \hat{a}) = 0 \Rightarrow \hat{a}(x) = \frac{\sum_{i=1}^n w_i(x) y_i}{\sum_{i=1}^n w_i(x)}.$$

Это локально–взвешенное среднее (в точности формула из метода скользящего окна).

Локально–линейная регрессия: аппроксимируем функцию в окрестности x линейно: $f(z) \approx a + b^\top (z - x)$. Решаем взвешенный МНК

$$(\hat{a}(x), \hat{b}(x)) = \arg \min_{a,b} \sum_{i=1}^n w_i(x) (y_i - a - b^\top (x_i - x))^2.$$

В матричном виде: пусть $Z = \begin{pmatrix} 1 & (x_1 - x)^\top \\ \vdots & \vdots \\ 1 & (x_n - x)^\top \end{pmatrix} \in \mathbb{R}^{n \times (1+m)}$, $W = \text{diag}(w_1(x), \dots, w_n(x))$, $y = (y_1, \dots, y_n)^\top$. Тогда решение (нормальные уравнения) есть

$$\begin{pmatrix} \hat{a}(x) \\ \hat{b}(x) \end{pmatrix} = (Z^\top W Z)^{-1} Z^\top W y, \quad \hat{f}(x) = \hat{a}(x).$$

Сравнение (кратко по смыслу лекций):

- **Локально–постоянная:** проще, меньше параметров; но имеет больший *смещённый* прогноз (bias), особенно на границах области и при неоднородной плотности X .
- **Локально–линейная:** учитывает локальный наклон, обычно уменьшает bias (в т.ч. на границах), но дисперсия выше (оценивать нужно и b).
- Обе зависят от выбора окна h (bias–variance компромисс) и ядра K .

29 Опишите деревья принятия решений, опишите алгоритм использования. Опишите алгоритм построения дерева, оцените его сложность.

Дерево решений (Decision Tree): модель вида «ветвления по признакам», где

- во *внутренних узлах* проверяется условие по одному признаку (обычно $x_j \leq t$),
- по результату идём в левое/правое поддерево,
- в *листье* хранится ответ: класс (классификация) или число (регрессия).

Алгоритм использования (предсказание): для объекта x стартуем из корня и последовательно выполняем тесты в узлах до листа.

- **Классификация:** в листе обычно берут $\hat{y} = \arg \max_c \hat{p}(c \mid \text{leaf})$ (большинство/частоты классов).
- **Регрессия:** в листе берут среднее/медиану \hat{y} по объектам, попавшим в лист.

Сложность предсказания: $O(\text{depth})$.

Построение дерева (жадный top-down, рекурсивное разбиение): в каждом узле S выбираем разбиение, которое максимально уменьшает «нечистоту» (impurity).

1. Для каждого признака j и порога t рассматриваем сплит:

$$S_L(j, t) = \{i \in S : x_{ij} \leq t\}, \quad S_R(j, t) = S \setminus S_L(j, t).$$

2. Считаем критерий качества сплита через уменьшение нечистоты:

$$\Delta(j, t) = I(S) - \frac{|S_L|}{|S|} I(S_L) - \frac{|S_R|}{|S|} I(S_R).$$

3. Выбираем $(j^*, t^*) = \arg \max \Delta(j, t)$, делим узел на два и повторяем рекурсивно.
4. Остановка (как в лекциях): узел «чистый», достигнут `max_depth`, мало объектов (`min_samples_leaf/split`), либо выигрыш Δ слишком мал.

(Конкретные формулы $I(\cdot)$: энтропия/Джини для классификации, дисперсия/МНК для регрессии — в следующем вопросе.)

Оценка сложности: пусть n — число объектов, m — число признаков.

a) **Обучение (типично):** для каждого узла перебираем признаки и пороги. Если по каждому признаку пороги берутся по отсортированным значениям, то суммарно часто оценивают как

$$O(m n \log n)$$

(за счёт сортировок) и умножают на константу, зависящую от реализации. В худшем случае при крайне несбалансированном дереве глубина может быть $O(n)$.

b) **Память:** хранение структуры дерева $O(\#nodes)$.

30 Дайте определение информативности. Выведите ее формулы из минимизации функции потерь — абсолютной, бинарной, квадратичной, логарифма правдоподобия.

Информативность (information gain) разбиения: в деревьях в каждом узле S вводят меру “нечистоты” $I(S)$ как минимальный эмпирический риск при константном предсказании в этом узле. Для сплита $S \rightarrow (S_L, S_R)$ информативность (выигрыш) определяют как уменьшение нечистоты:

$$\text{Gain}(S \rightarrow S_L, S_R) = I(S) - \frac{|S_L|}{|S|} I(S_L) - \frac{|S_R|}{|S|} I(S_R).$$

Общий принцип вывода $I(S)$ из минимизации потерь:

$$I(S) = \min_{\theta} \frac{1}{|S|} \sum_{i \in S} \ell(y_i, \theta),$$

где θ — параметр константного ответа в листе (число a , класс c , вероятности p).

1) **Абсолютная потеря (регрессия):** $\ell(y, a) = |y - a|$.

$$\hat{a} = \arg \min_a \sum_{i \in S} |y_i - a| \Rightarrow \hat{a} = \text{med}(\{y_i\}_{i \in S}).$$

Тогда

$$I(S) = \frac{1}{|S|} \sum_{i \in S} |y_i - \text{med}(y)|.$$

2) **Квадратичная потеря (регрессия):** $\ell(y, a) = (y - a)^2$.

$$\hat{a} = \arg \min_a \sum_{i \in S} (y_i - a)^2 \Rightarrow \hat{a} = \bar{y}_S := \frac{1}{|S|} \sum_{i \in S} y_i.$$

Тогда

$$I(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y}_S)^2 \quad (= \widehat{\text{Var}}_S(Y) \text{ с точностью до делителя}).$$

3) Бинарная (0–1) потеря (классификация): $\ell(y, c) = \mathbb{I}\{y \neq c\}$.

$$\hat{c} = \arg \min_c \sum_{i \in S} \mathbb{I}\{y_i \neq c\} \Rightarrow \hat{c} = \arg \max_c n_c,$$

где $n_c = \#\{i \in S : y_i = c\}$. Если $p_c = n_c/|S|$, то

$$I(S) = \min_c (1 - p_c) = 1 - \max_c p_c.$$

4) Логарифм правдоподобия (log-loss / кросс-энтропия): в листе предсказываем вероятности классов $p = (p_1, \dots, p_K)$, $\sum_c p_c = 1$, и берём $\ell(y, p) = -\log p_y$. Тогда

$$\hat{p} = \arg \min_p \frac{1}{|S|} \sum_{i \in S} -\log p_{y_i} \Rightarrow \hat{p}_c = \frac{n_c}{|S|}.$$

Минимальное значение потерь равно энтропии эмпирического распределения классов:

$$I(S) = - \sum_{c=1}^K p_c \log p_c \quad (\text{в бинарном случае } -p \log p - (1-p) \log(1-p)).$$

31 Выведите формулу Bias–Variance trade–off для задачи регрессии. Покажите уменьшение дисперсии при применении пастинга.

Bias–Variance trade–off (регрессия, квадратичная потеря): пусть данные порождены как

$$Y = f(X) + \varepsilon, \quad \mathbb{E}[\varepsilon | X] = 0, \quad \text{Var}(\varepsilon | X) = \sigma^2.$$

Пусть $\hat{f}_D(x)$ — предсказатель, обученный на случайной выборке D . Тогда при фиксированном x имеем разложение ожидаемой ошибки:

$$\mathbb{E}_{D,\varepsilon}[(Y - \hat{f}_D(x))^2 | X = x] = \underbrace{(\text{Bias}(x))^2}_{\text{смещение}} + \underbrace{\text{Var}(x)}_{\text{дисперсия}} + \underbrace{\sigma^2}_{\text{шум}}.$$

Где

$$\text{Bias}(x) = \mathbb{E}_D[\hat{f}_D(x)] - f(x), \quad \text{Var}(x) = \mathbb{E}_D[(\hat{f}_D(x) - \mathbb{E}_D \hat{f}_D(x))^2].$$

(Интегрируя по распределению X получаем такое же разложение для риска $\mathbb{E}[(Y - \hat{f}(X))^2]$.)

Пастинг (pasting): уменьшение дисперсии при усреднении. Пастинг в лекциях: обучаем M моделей на разных подвыборках (обычно без возвращения), а предсказание берём усреднением $\bar{f}(x) = \frac{1}{M} \sum_{m=1}^M \hat{f}^{(m)}(x)$. Тогда

$$\text{Var}(\bar{f}(x)) = \frac{1}{M^2} \sum_{m=1}^M \text{Var}(\hat{f}^{(m)}(x)) + \frac{2}{M^2} \sum_{m < \ell} \text{Cov}(\hat{f}^{(m)}(x), \hat{f}^{(\ell)}(x)).$$

Если модели одинаковы по дисперсии $\text{Var}(\hat{f}^{(m)}(x)) = v(x)$ и имеют попарную корреляцию $\rho(x)$, то

$$\text{Var}(\bar{f}(x)) = v(x) \left(\rho(x) + \frac{1 - \rho(x)}{M} \right) \leq v(x),$$

а при независимости ($\rho = 0$) получаем $\text{Var}(\bar{f}(x)) = v(x)/M$. Смещение при простом усреднении не увеличивается (для $\mathbb{E}_D \hat{f}$ оно остаётся тем же), поэтому пастинг в первую очередь снижает дисперсию.

32 Опишите метод бэггинга. Приведите аргументы в пользу его обоснования.

Бэггинг (bagging = bootstrap aggregating): ансамбль, где базовые модели обучаются на бутстрэп-подвыборках и затем агрегируют ответы.

Алгоритм (как в лекциях):

t) Пусть есть обучающая выборка $D = \{(x_i, y_i)\}_{i=1}^n$ и базовый алгоритм $a(\cdot)$.

1) Для $m = 1, \dots, M$:

- Сформировать бутстрэп-выборку $D^{(m)}$ размера n с *возвращением* из D .
- Обучить базовую модель $\hat{f}^{(m)} = a(D^{(m)})$.

2) Агрегация предсказаний:

- **Регрессия:** $\bar{f}(x) = \frac{1}{M} \sum_{m=1}^M \hat{f}^{(m)}(x)$.
- **Классификация:** либо голосование $\hat{y}(x) = \arg \max_c \sum_{m=1}^M \mathbb{I}\{\hat{y}^{(m)}(x) = c\}$, либо усреднение вероятностей $\bar{p}_c(x) = \frac{1}{M} \sum_m \hat{p}_c^{(m)}(x)$ и $\arg \max_c \bar{p}_c(x)$.

Аргументы обоснования (по сути лекций):

- **Снижение дисперсии:** это усреднение коррелированных предсказателей. При $\text{Var}(\hat{f}^{(m)}(x)) = v(x)$ и попарной корреляции $\rho(x)$:

$$\text{Var}(\bar{f}(x)) = v(x) \left(\rho(x) + \frac{1 - \rho(x)}{M} \right) \leq v(x),$$

а при $\rho \approx 0$ получаем почти v/M .

- **Смещение обычно почти не растёт:** агрегация (простое среднее) меняет в основном дисперсию, поэтому бэггинг особенно полезен для *нестабильных* алгоритмов (деревья), где дисперсия велика.
- **Bootstrap даёт разнообразие моделей:** каждая $D^{(m)}$ отличается; в среднем уникальных объектов в бутстрэпе около $1 - e^{-1} \approx 0.632$ от n (остальные повторяются).
- **ООВ-оценка качества:** объекты, не попавшие в $D^{(m)}$ (out-of-bag), можно использовать для валидации без отдельного hold-out.

33 Опишите алгоритм построения случайного леса. Как выбирать параметры леса?

Случайный лес (Random Forest): ансамбль деревьев, который сочетает идеи бэггинга (bootstrap) и дополнительной рандомизации при выборе признаков в узлах. Цель по лекциям: снизить корреляцию деревьев (ρ) и тем самым уменьшить дисперсию ансамбля при усреднении.

Алгоритм построения (кратко, по шагам): пусть обучающая выборка $D = \{(x_i, y_i)\}_{i=1}^n$, число деревьев M .

1) Для $m = 1, \dots, M$:

- Сформировать бутстрэп-выборку $D^{(m)}$ размера n с *возвращением* (как в бэггинге).
- Обучить дерево решений на $D^{(m)}$, но в каждом узле:
 - случайно выбрать подмножество признаков \mathcal{J} размера m_{try} ;
 - искать лучший сплит (максимум информативности / минимум нечистоты) *только* среди признаков из \mathcal{J} .
- Обычно деревья растят достаточно глубокими (малый bias), а variance снимают усреднением.

2) Агрегация ответов:

- **Регрессия:** $\bar{f}(x) = \frac{1}{M} \sum_{m=1}^M \hat{f}^{(m)}(x)$.
- **Классификация:** голосование большинства или усреднение вероятностей по деревьям.

3) **OOB (out-of-bag) оценка:** для каждого объекта i можно предсказывать его ответом только тех деревьев, где он *не* попал в бутстрэп. Это даёт оценку качества без отдельной валидационной выборки.

Как выбирать параметры леса (типовая логика лекций):

- **Число деревьев M (n_estimators):** увеличиваем, пока качество по OOB/CV не перестаёт расти. Больше деревьев обычно снижает дисперсию, но даёт линейный рост времени.
- **Число признаков в узле m_{try} (max_features):** ключевой параметр, управляющий корреляцией деревьев. Меньше $m_{\text{try}} \Rightarrow$ меньше корреляция, но слабее отдельные деревья. (Частое правило из лекций: для классификации \sqrt{p} , для регрессии $p/3$, где p — число признаков.)
- **Сложность деревьев (max_depth, min_samples_leaf / split):** глубже дерево \Rightarrow меньше bias, но больше variance. В RF обычно допускают глубокие деревья, но при шуме/малом n полезно ограничивать глубину или ставить min_samples_leaf.
- **Bootstrap / размер подвыборки (bootstrap, max_samples):** стандартно bootstrap включён; при уменьшении max_samples можно усилить разнообразие (но сделать деревья слабее).

- **Прочее:** критерий сплита (Gini/entropy, MSE/MAE), балансировка классов (class_weight) — подбирают по задаче и метрике.

34 Опишите алгоритмы стекинга и блендинга. Как находить и использовать feature importance?

Стекинг (stacking): обучение нескольких моделей и использование их предсказаний как признаков для *мета-модели*. В лекциях: отличие от бэггинга в том, что агрегирование делается *другим алгориттом*, а не простым усреднением.

Алгоритм стекинга (как в лекциях):

1. Разбить выборку на K частей: $D = D_1 \cup \dots \cup D_K$.
2. Обучить K моделей одного типа a_1, \dots, a_K (например, деревья), где a_k обучается на $D \setminus D_k$.
3. Для каждого k получить предсказания на отложенной части D_k (то есть на данных, не использованных при обучении a_k).
4. Обучить мета-модель b на парах

$$b = a\left(\{(\hat{y}_i, y_i)\}_{i=1}^n\right), \quad \hat{y}_i \text{ — предсказание базовой модели для } x_i.$$

5. Для тестового объекта x подать в b агрегированное предсказание базовых моделей:

$$\hat{y}(x) = b\left(\frac{1}{K} \sum_{k=1}^K a_k(x)\right).$$

Блендинг (blending): в лекциях отмечено, что он отличается от стекинга тем, что *семплирование (разбиение на подвыборки)* не делается, и на каждую модель подаются одни и те же данные.

Feature importance (важность признаков) для деревьев/леса: показывает вклад признака в предсказание; в лекциях задаётся через сумму уменьшений нечистоты по узлам. Для узла v :

$$\text{Imp}_v = \frac{|Q_v|}{|Q|} \left(I(Q_v) - \frac{|Q_{v1}|}{|Q_v|} I(Q_{v1}) - \frac{|Q_{v2}|}{|Q_v|} I(Q_{v2}) \right),$$

где Q_v — множество объектов, попавших в узел v , Q_{v1}, Q_{v2} — множества в его потомках после разбиения, $I(\cdot)$ — нечистота (критерий) в узле.

Пусть V_j — множество узлов, где при разбиении использовался признак j ($j = 1, \dots, p$). Тогда нормированная важность:

$$R_j = \frac{\sum_{v \in V_j} \text{Imp}_v}{\sum_{k=1}^p \sum_{v \in V_k} \text{Imp}_v}, \quad \sum_{j=1}^p R_j = 1.$$

Для случайного леса из M деревьев усредняют по деревьям:

$$\bar{R}_j = \frac{1}{M} \sum_{m=1}^M R_{mj}.$$

35 Опишите алгоритм градиентного бустинга.

Градиентный бустинг (gradient boosting): построение композиции (ансамбля) слабых алгоритмов в виде суммы

$$F_M(x) = \sum_{m=0}^M \gamma_m h_m(x),$$

где на каждом шаге добавляется новая базовая модель h_m так, чтобы уменьшить выбранную функцию потерь.

Алгоритм (в терминах лекций: шаги функционального градиентного спуска): пусть задана дифференцируемая по F потеря $\ell(y, F(x))$.

1. **Инициализация:** выбрать константное приближение

$$F_0(x) = \arg \min_{c \in \mathbb{R}} \sum_{i=1}^n \ell(y_i, c).$$

2. Для $m = 1, \dots, M$:

(a) **Псевдо-остатки (антиградиент):**

$$r_i^{(m)} = - \left. \frac{\partial \ell(y_i, F(x_i))}{\partial F} \right|_{F=F_{m-1}}, \quad i = 1, \dots, n.$$

(b) **Обучить базовый алгоритм** h_m на выборке $\{(x_i, r_i^{(m)})\}$ (т.е. аппроксимировать антиградиент).

(c) **Подбор шага** (линейный поиск):

$$\gamma_m = \arg \min_{\gamma \in \mathbb{R}} \sum_{i=1}^n \ell(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

(d) **Обновление композиции:**

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

где $\eta \in (0, 1]$ — скорость обучения (shrinkage).

Замечания (кратко): обычно берут h_m в виде небольшого дерева решений; регуляризация достигается малыми деревьями, малым η и ограничением числа итераций M .

36 Выведите формулы градиентного бустинга на решающих деревьях для квадратичной и логистической функции потерь. Приведите примеры методов ускорения бустинга.

Перцептрон Розенблатта (однослойный): бинарный линейный классификатор

$$a(x) = w^\top x + b, \quad \hat{y} = \text{sign}(a(x)), \quad y \in \{-1, +1\}.$$

(Эквивалентно: $\hat{y} = \mathbb{I}\{a(x) \geq 0\}$ при $y \in \{0, 1\}\text{.}$)

Правило обучения (perceptron learning rule, как в лекциях): идём по объектам и исправляем веса только при ошибке. Если $y_i a(x_i) \leq 0$ (объект классифицирован неверно), то

$$w \leftarrow w + \eta y_i x_i, \quad b \leftarrow b + \eta y_i,$$

где $\eta > 0$ — шаг. Если классификация верна ($y_i a(x_i) > 0$), то обновления нет.

Свойство: алгоритм сходится за конечное число шагов, если выборка линейно разделима (теорема о сходимости перцептрона).

Примеры других функций активации:

- пороговая (Heaviside): $\sigma(a) = \mathbb{I}\{a \geq 0\};$
- сигмоида: $\sigma(a) = \frac{1}{1 + e^{-a}};$
- \tanh : $\sigma(a) = \tanh(a);$
- ReLU: $\sigma(a) = \max\{0, a\};$
- leaky ReLU: $\sigma(a) = \max\{\alpha a, a\}, \alpha \in (0, 1).$

37 Опишите перцепtron Розенблатта. Приведите примеры других функций активации.

Перцепtron Розенблатта (однослойный): бинарный линейный классификатор

$$a(x) = w^\top x + b, \quad \hat{y} = \text{sign}(a(x)), \quad y \in \{-1, +1\}.$$

(Эквивалентно: $\hat{y} = \mathbb{I}\{a(x) \geq 0\}$ при $y \in \{0, 1\}\text{.}$)

Правило обучения (perceptron learning rule, как в лекциях): обновляем параметры только при ошибке. Если $y_i a(x_i) \leq 0$, то

$$w \leftarrow w + \eta y_i x_i, \quad b \leftarrow b + \eta y_i,$$

где $\eta > 0$ — шаг. Если $y_i a(x_i) > 0$, то обновления нет.

Свойство: при линейной разделимости обучающей выборки алгоритм сходится за конечное число шагов (теорема о сходимости перцептрона).

Примеры других функций активации:

- пороговая (Heaviside): $\sigma(a) = \mathbb{I}\{a \geq 0\};$
- сигмоида: $\sigma(a) = \frac{1}{1 + e^{-a}};$
- \tanh : $\sigma(a) = \tanh(a);$
- ReLU: $\sigma(a) = \max\{0, a\};$
- leaky ReLU: $\sigma(a) = \max\{\alpha a, a\}, \alpha \in (0, 1).$

38 Дайте определение нейронной сети. Сформулируйте теорему Цыбенко.

Нейронная сеть (feed-forward, многослойный перцептрон): параметрическое отображение $f_\theta : \mathbb{R}^m \rightarrow \mathbb{R}^q$, построенное как композиция слоёв вида

$$h^{(0)} = x, \quad h^{(\ell)} = \sigma(W^{(\ell)}h^{(\ell-1)} + b^{(\ell)}), \quad \ell = 1, \dots, L-1,$$

$$f_\theta(x) = W^{(L)}h^{(L-1)} + b^{(L)} \quad (\text{или } \sigma \text{ на выходе, в зависимости от задачи}).$$

Здесь $W^{(\ell)}, b^{(\ell)}$ — обучаемые параметры, σ — нелинейная функция активации.

Теорема Цыбенко (универсальная аппроксимация, 1989): пусть $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ — сигмоидальная функция (ограниченная, измеримая и такая, что $\lim_{t \rightarrow -\infty} \sigma(t) = 0$, $\lim_{t \rightarrow +\infty} \sigma(t) = 1$). Тогда для любого компактного множества $K \subset \mathbb{R}^m$ и любой непрерывной функции $f \in C(K)$, для любого $\varepsilon > 0$ существует односкрытослойная сеть

$$F(x) = \sum_{j=1}^N a_j \sigma(w_j^\top x + b_j)$$

такая, что

$$\sup_{x \in K} |F(x) - f(x)| < \varepsilon.$$

То есть сеть с *одним скрытым слоем* и сигмоидальной активацией может сколь угодно точно аппроксимировать любую непрерывную функцию на компакте.

39 Выведите формулы метода обратного распространения ошибок на примере сети с одним скрытым слоем. Сформулируйте общий принцип.

Сеть с одним скрытым слоем (векторная запись): пусть $x \in \mathbb{R}^m$, скрытый слой размера h , выход $\hat{y} \in \mathbb{R}^q$.

$$\begin{aligned} z^{(1)} &= W^{(1)}x + b^{(1)}, & a^{(1)} &= \sigma(z^{(1)}), \\ z^{(2)} &= W^{(2)}a^{(1)} + b^{(2)}, & \hat{y} &= \varphi(z^{(2)}). \end{aligned}$$

Функция потерь на одном объекте: $\mathcal{L} = \ell(\hat{y}, y)$.

Backprop (цепное правило): вводим “ошибки” (дельты)

$$\delta^{(2)} := \frac{\partial \mathcal{L}}{\partial z^{(2)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \odot \varphi'(z^{(2)}),$$

где \odot — покомпонентное умножение. Тогда градиенты по параметрам выходного слоя:

$$\frac{\partial \mathcal{L}}{\partial W^{(2)}} = \delta^{(2)} (a^{(1)})^\top, \quad \frac{\partial \mathcal{L}}{\partial b^{(2)}} = \delta^{(2)}.$$

Далее ошибка переносится на скрытый слой:

$$\delta^{(1)} := \frac{\partial \mathcal{L}}{\partial z^{(1)}} = ((W^{(2)})^\top \delta^{(2)}) \odot \sigma'(z^{(1)}).$$

И градиенты по параметрам первого слоя:

$$\frac{\partial \mathcal{L}}{\partial W^{(1)}} = \delta^{(1)} x^\top, \quad \frac{\partial \mathcal{L}}{\partial b^{(1)}} = \delta^{(1)}.$$

Общий принцип (для L слоёв): если $z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)}$, $a^{(\ell)} = \sigma(z^{(\ell)})$, то

$$\delta^{(L)} = \frac{\partial \mathcal{L}}{\partial z^{(L)}}, \quad \delta^{(\ell)} = ((W^{(\ell+1)})^\top \delta^{(\ell+1)}) \odot \sigma'(z^{(\ell)}), \quad \ell = L-1, \dots, 1,$$

а градиенты считаются локально в каждом слое:

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \delta^{(\ell)} (a^{(\ell-1)})^\top, \quad \frac{\partial \mathcal{L}}{\partial b^{(\ell)}} = \delta^{(\ell)}.$$

(Для мини-батча градиенты усредняют по объектам.)

40 Опишите способы инициализации Ксавье и Kaiming. Приведите их математический вывод.

Цель инициализации (по лекциям): выбрать дисперсию весов так, чтобы при прямом и обратном проходе не происходило взрыва/затухания дисперсий сигналов. Пусть в слое

$$z = Wx, \quad x \in \mathbb{R}^{n_{\text{in}}}, \quad z \in \mathbb{R}^{n_{\text{out}}},$$

где элементы x_i независимы, $\mathbb{E}[x_i] = 0$, $\text{Var}(x_i) = q$, а веса w_{ij} независимы, $\mathbb{E}[w_{ij}] = 0$, $\text{Var}(w_{ij}) = s$.

Выход условия по прямому проходу. Для компоненты $z_j = \sum_{i=1}^{n_{\text{in}}} w_{ji} x_i$ имеем

$$\text{Var}(z_j) = \sum_{i=1}^{n_{\text{in}}} \text{Var}(w_{ji} x_i) = n_{\text{in}} s q.$$

Чтобы сохранялась дисперсия (в среднем) $\text{Var}(z_j) \approx q$, нужно

$$n_{\text{in}} s q = q \Rightarrow s = \frac{1}{n_{\text{in}}}.$$

Выход условия по обратному проходу. Пусть $\delta = \partial \mathcal{L} / \partial z$ (градиент по предактивациям). Тогда для предыдущего слоя

$$\delta^{\text{prev}} = W^\top \delta \quad (\text{без учёта производной активации, как в стандартном выводе}).$$

Аналогично получаем

$$\text{Var}(\delta_i^{\text{prev}}) = n_{\text{out}} s \text{Var}(\delta_j).$$

Чтобы дисперсия градиента сохранялась, нужно $s = 1/n_{\text{out}}$.

Инициализация Ксавье (Glorot/Xavier). Компромисс между требованиями прямого и обратного проходов берут как среднее:

$$\text{Var}(w_{ij}) = s = \frac{2}{n_{\text{in}} + n_{\text{out}}}.$$

Два стандартных варианта (для w_{ij}):

- **Xavier normal:** $w_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}} + n_{\text{out}}}\right)$.
- **Xavier uniform:** $w_{ij} \sim \mathcal{U}[-a, a]$, где из $\text{Var}(\mathcal{U}[-a, a]) = a^2/3$ получаем

$$a = \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}.$$

(В лекциях: обычно применяют для сигмоиды/tanh и «мягких» активаций.)

Инициализация Kaiming (He). Для ReLU $\sigma(u) = \max\{0, u\}$ при симметричном u вокруг нуля верно приближение

$$\text{Var}(\sigma(u)) \approx \frac{1}{2} \text{Var}(u).$$

Чтобы после ReLU дисперсия сохранялась, нужно компенсировать фактор 1/2:

$$\text{Var}(z) = n_{\text{in}} s q \Rightarrow \text{Var}(\sigma(z)) \approx \frac{1}{2} n_{\text{in}} s q \stackrel{!}{=} q \Rightarrow s = \frac{2}{n_{\text{in}}}.$$

Стандартные варианты:

- **He normal:** $w_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$.
- **He uniform:** $w_{ij} \sim \mathcal{U}[-a, a]$, где

$$a = \sqrt{\frac{6}{n_{\text{in}}}}.$$

Обобщение для leaky ReLU с наклоном α на отрицательной части (часто дают в лекциях):

$$\text{Var}(w) = \frac{2}{(1 + \alpha^2) n_{\text{in}}}.$$

Замечание: n_{in} и n_{out} — это *fan_in* и *fan_out* (числа входов/выходов наблюдаемой нейронной связи; для свёрток берут ещё размер ядра).

41 Опишите прием Dropout. Дайте его математическое обоснование.

Dropout (лекционный смысл): регуляризация нейросети случайным *занулением* части нейронов (активаций) на обучении. Идея: заставить сеть не полагаться на отдельные «удачные» признаки и уменьшить переобучение.

Формализация (inverted dropout): пусть для некоторого слоя получены активации $h \in \mathbb{R}^d$. На обучении генерируем маску Бернулли

$$m \sim \text{Bernoulli}(p)^d, \quad m_j \in \{0, 1\}, \quad \mathbb{P}(m_j = 1) = p,$$

и используем запущённые активации

$$\tilde{h} = \frac{m \odot h}{p},$$

где \odot — покомпонентное умножение, p — вероятность «сохранения» нейрона. На тесте dropout не применяется: берут $\tilde{h} = h$.

Почему делим на p (сохранение матожидания):

$$\mathbb{E}[\tilde{h}_j] = \mathbb{E}\left[\frac{m_j h_j}{p}\right] = \frac{h_j}{p} \mathbb{E}[m_j] = \frac{h_j}{p} p = h_j.$$

То есть в среднем слой на обучении «видит» те же масштабы, что и на тесте.

Математическое обоснование (как в лекциях): обучение с dropout — это минимизация ожидаемой функции потерь по случайной маске:

$$\min_{\theta} \mathbb{E}_m \left[\frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\theta}(x_i; m)) \right].$$

Интерпретация: это приближённое усреднение по экспоненциальному числу под-сетей (каждая маска m задаёт свою «подмодель»), т.е. *model averaging*. За счёт этого уменьшается дисперсия предсказателя и улучшается обобщение.

Короткое замечание про эффект регуляризации: в простых линейных/логистических моделях усреднение по бернульевскому мультиплексивному шуму даёт добавку к целевой функции, ведущую себя как штраф на веса (эффект близок к ℓ_2 -регуляризации). (На уровне лекционной интуиции: dropout добавляет шум в признаки/активации, и модель вынуждена делать решения устойчивыми к этому шуму.)

Практические настройки (минимум): обычно применяют к скрытым слоям; типичные p в лекциях — порядка 0.5 для полно связанных слоёв и ближе к 0.8–0.9 для свёрточных.

42 Опишите приемы изменения скорости обучения и нормализации для обучения нейросетей.

Изменение скорости обучения (learning rate). Градиентные методы обновляют параметры так:

$$\theta_{t+1} = \theta_t - \eta_t g_t, \quad g_t = \nabla_{\theta} \mathcal{L}(\theta_t),$$

где η_t можно делать зависящей от шага t : большой шаг в начале (быстрое обучение), малый в конце (стабильная сходимость).

Типичные расписания η_t :

- *Step decay:* $\eta_t = \eta_0 \gamma^{\lfloor t/s \rfloor}$ (каждые s эпохи умножаем на $\gamma < 1$).
- *Экспоненциальный спад:* $\eta_t = \eta_0 e^{-kt}$.
- *Гиперболический спад:* $\eta_t = \frac{\eta_0}{1 + kt}$.
- *Cosine annealing:*

$$\eta_t = \eta_{\min} + \frac{\eta_{\max} - \eta_{\min}}{2} (1 + \cos(\pi t / T)).$$

- *Warmup*: первые T_w шагов η_t линейно растёт до η_{\max} (часто помогает при больших батчах/Adam).
- *Warm restarts (SGDR)*: cosine-расписание периодически “перезапускают”.

Адаптивные методы как “покоординатная” настройка шага:

- *AdaGrad*: $G_t = \sum_{\tau=1}^t g_\tau \odot g_\tau$, $\theta \leftarrow \theta - \eta \frac{g_t}{\sqrt{G_t} + \varepsilon}$.
 - *RMSProp*: $v_t = \beta v_{t-1} + (1 - \beta)g_t^2$, $\theta \leftarrow \theta - \eta \frac{g_t}{\sqrt{v_t} + \varepsilon}$.
 - *Adam*: $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$, $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$, с bias-correction \hat{m}_t, \hat{v}_t :
- $$\theta \leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

Нормализация (зачем): стабилизирует масштабы/распределения активаций и градиентов, обычно позволяет брать больший η , ускоряет и делает обучение устойчивее.

Нормализация входов: стандартизация признаков

$$x \leftarrow \frac{x - \mu}{\sigma} \quad (\text{часто по каждому признаку/каналу}).$$

Batch Normalization (BN). Для активаций x внутри мини-батча B :

$$\mu_B = \frac{1}{|B|} \sum_{i \in B} x_i, \quad \sigma_B^2 = \frac{1}{|B|} \sum_{i \in B} (x_i - \mu_B)^2,$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \quad y_i = \gamma \hat{x}_i + \beta.$$

На инференсе используют скользящие оценки μ, σ^2 (накопленные при обучении).

LayerNorm / GroupNorm (когда батч маленький).

- *LayerNorm*: нормируем внутри одного объекта по признакам (не зависит от размера батча).
- *GroupNorm*: нормируем по группам каналов (компромисс для CNN).