ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Лекция 11

Автоэнкодеры и вариационные автоэнкодеры

Представления данных, снижение размерности, обнаружение аномалий и генеративное моделирование

Преподаватель: Аршын Алтыбай
Кафедра компьютерной инженерии,
Алматинский технологический университет, Алматы, Казахстан
Учебный год: 2026

> **Связь с предыдущими лекциями.** В лекциях 1–10 были рассмотрены перцептрон, MLP, алгоритм обратного распространения, оптимизация, регуляризация, CNN и рекуррентные сети. Теперь исследуем обучение представлений без обязательного использования меток классов. Автоэнкодер учится восстанавливать вход, а вариационный автоэнкодер задаёт вероятностную генеративную модель. Основные ссылки: [6, 3, 4, 10].

1. Цель, предварительные знания и ожидаемые результаты

Цель лекции — получить математически корректное понимание автоэнкодеров, способов их обучения, ограничений и практических применений, вывести вариационную нижнюю оценку правдоподобия (ELBO), реализовать модели на NumPy и PyTorch и научиться проектировать воспроизводимые эксперименты.

Предварительные знания: умножение матриц, производная сложной функции, градиентный спуск, понятие вероятности и условной плотности, основные механизмы MLP и CNN. Для разделов про VAE полезны распределение Гаусса, математическое ожидание и дисперсия; необходимые формулы будут повторены.

После изучения студент сможет:

1. записать отображения кодировщика и декодировщика и проверить размерности;

2. объяснить роль bottleneck и почему восстановление входа не равно пониманию данных;
3. вывести градиенты для простого линейного автоэнкодера;
4. сравнить линейный AE с методом главных компонент (PCA);
5. выбрать MSE, BCE или другую функцию потерь обоснованно;
6. объяснить и реализовать разреженный, шумоподавляющий и свёрточный AE;
7. настроить детектор аномалий без подгонки порога по тестовым данным;
8. сформулировать вероятностную модель $p_\theta(x, z)$ и ELBO;
9. вычислить KL-дивергенцию гауссовского апостериорного приближения;
10. применить трюк репараметризации для дифференцируемого обучения;
11. сравнить детерминированный AE и VAE на реальных данных;
12. критически оценить ограничения реконструкционной ошибки и генеративных моделей.

План: постановка задачи; линейные и глубокие AE; реконструкция и регуляризация; методы обнаружения аномалий; вероятностная модель VAE; ELBO и репараметризация; код и эксперименты; контрольные вопросы.

2. Обучение представлений: почему одних меток недостаточно

2.1 Обучение с учителем и без учителя

При *обучении с учителем* имеется набор $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, где $x_i \in \mathbb{R}^d$ — объект, y_i — метка. Например, CNN учится предсказывать класс изображения. При *обучении без учителя* доступны только объекты $\{x_i\}_{i=1}^N$, а интерес представляют структура данных, компактные признаки, распределение объектов и нетипичные наблюдения. Автоэнкодеры часто рассматривают как модели обучения представлений без учителя; точнее, они используют *самоконтролируемый сигнал*: целевой результат получается непосредственно из самого входа [6, 4].

Рассмотрим изображения размером 28×28 пикселей: каждое можно преобразовать в вектор $x \in [0,1]^{784}$. Хотя формально пространство имеет размерность 784, реальные изображения рукописных цифр могут обладать низкоразмерной структурой: наклон, толщина штриха, положение центра, характер написания. Возникает задача найти представление $z \in \mathbb{R}^k$ с $k \ll 784$, сохраняющее информацию, необходимую для восстановления типичных объектов.

2.2 Представление и восстановление

Пусть $f_\phi : \mathbb{R}^d \to \mathbb{R}^k$ — *кодировщик* (encoder), $g_\theta : \mathbb{R}^k \to \mathbb{R}^d$ — *декодировщик* (decoder). Тогда

$$z = f_\phi(x), \qquad \widehat{x} = g_\theta(z) = g_\theta(f_\phi(x)). \tag{1}$$

Параметры ϕ и θ оптимизируются так, чтобы $\widehat{x}$ было близко к x. Код z называют *латентным представлением* (latent representation), а промежуточный узкий слой — *бутылочным горлышком* (bottleneck).

> **Принципиальное различие** Цель реконструкции не гарантирует, что координаты z интерпретируются человеком или оптимальны для классификации. AE может научиться сохранять детали, незначимые для целевой задачи, и игнорировать редкие, но важные признаки. Представления следует оценивать на конкретной задаче, а не только по малой ошибке восстановления.

2.3 Сценарии применения

Сжатие данных без гарантии оптимальности кодирования; предварительное обучение признаков; денойзинг; выделение нелинейных низкоразмерных многообразий; выявление нетипичных образцов; генерация объектов через вероятностные варианты. Для медицины, производства и информационной безопасности требуется отдельная проверка надёжности, потому что низкая ошибка восстановления не всегда означает нормальность объекта.

3. Архитектура базового автоэнкодера

3.1 Полносвязный автоэнкодер

В простейшем случае кодировщик и декодировщик состоят из последовательностей аффинных отображений и функций активации:

$$h^{(1)} = \sigma_1(W_1 x + b_1), \qquad z = \sigma_2(W_2 h^{(1)} + b_2), \tag{2}$$

$$h^{(3)} = \sigma_3(W_3 z + b_3), \qquad \widehat{x} = \sigma_4(W_4 h^{(3)} + b_4). \tag{3}$$

Если $d = 784$ и архитектура задана как $784 \to 256 \to 32 \to 256 \to 784$, то $z \in \mathbb{R}^{32}$. Весовые матрицы имеют размеры 256×784, 32×256, 256×32, 784×256 соответственно.

Суммарное число обучаемых параметров полного полносвязного AE определяется формулой

$$P = \sum_{\ell=1}^{L} (n_{\ell-1} n_\ell + n_\ell), \tag{4}$$

где n_ℓ — число нейронов на слое, включая выходной. Для указанной архитектуры:

$$P = (784 \cdot 256 + 256) + (256 \cdot 32 + 32) + (32 \cdot 256 + 256) + (256 \cdot 784 + 784).$$

Точное вычисление выполняем отдельно: $200704+256 = 200960$, $8192+32 = 8224$, $8192 + 256 = 8448$, $200704 + 784 = 201488$, итого

$$\boxed{P = 419\,120.} \tag{5}$$

Заметим, что всего 32 координаты латентного вектора не означают 32 обучаемых параметра: большая часть параметров находится в матрицах кодировщика и декодировщика.

Слой	Вход	Выход	Параметры
Linear 1	784	256	200960
Linear 2	256	32	8224
Linear 3	32	256	8448
Linear 4	256	784	201488
Итого			**419120**

Таблица 1: Подсчёт параметров полносвязного автоэнкодера.

3.2 Почему нужны нелинейные функции активации

Если все слои линейны и отсутствуют ограничения, то композиция линейных отображений снова линейна: $\widehat{x} = W_D W_E x$. Сеть не способна представлять произвольные нелинейные отношения. Введение ReLU, GELU, tanh или другой нелинейности позволяет аппроксимировать более сложные отображения. Но слишком мощный декодировщик может компенсировать недостатки латентного кода; поэтому архитектуру следует подбирать вместе с задачей и регуляризацией [6].

3.3 Undercomplete и overcomplete autoencoders

Если $k < d$, AE называют *undercomplete*: размерность узкого представления меньше входной. Если $k \geq d$, AE *overcomplete*; в отсутствие дополнительных ограничений сеть способна приближать тождественное преобразование, не извлекая полезной структуры. Даже узкое представление не гарантирует хороших признаков при слишком большой мощности сети и небольшом датасете. Выход: разреженность, штрафы на чувствительность, добавление шума и другие явные ограничения.

4. Функции потерь и корректная интерпретация результата

4.1 Среднеквадратичная ошибка

Для вещественных признаков часто используют

$$\mathcal{L}_{\text{MSE}}(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{d} \|x_i - g_\theta(f_\phi(x_i))\|_2^2. \tag{6}$$

Деление на d — соглашение о масштабе; оно изменяет численное значение loss и масштаб градиента, но не положение минимума, если все остальные члены отсутствуют. Для одного объекта и скалярного $\widehat{x}$ производная $\partial(\widehat{x} - x)^2 / \partial \widehat{x} = 2(\widehat{x} - x)$.

4.2 Бинарная кросс-энтропия

Если компоненты входа интерпретируются как вероятности независимых бинарных событий, можно применять

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{Nd} \sum_{i,j} \left[x_{ij} \log \widehat{x}_{ij} + (1 - x_{ij}) \log(1 - \widehat{x}_{ij}) \right]. \tag{7}$$

В вычислениях предпочтительнее `BCEWithLogitsLoss`, которая принимает *логиты* и объединяет sigmoid с устойчивой формулой BCE. Входная нормировка $[0, 1]$ сама по себе не превращает вещественное изображение в истинную независимую Bernoulli-модель; выбор BCE должен соответствовать моделируемой величине. Для непрерывных пикселей часто применяют гауссовскую модель или иные распределения наблюдений.

4.3 Абсолютная и робастная ошибки

Сравним $L_1 = \sum_j |x_j - \widehat{x}_j|$, $L_2 = \sum_j (x_j - \widehat{x}_j)^2$ и Huber loss. Последняя квадратична для малых ошибок и линейна для больших. При наличии выбросов MSE чрезмерно усиливает вклад отдельных больших отклонений. Выбор метрики зависит от данных и целевой прикладной функции.

4.4 Численный пример выбора функции потерь

Пусть $x = (0, 1, 1)$, а $\widehat{x} = (0.1, 0.8, 0.6)$. Тогда средняя квадратичная ошибка равна

$$\frac{(0-0.1)^2 + (1-0.8)^2 + (1-0.6)^2}{3} = \frac{0.01 + 0.04 + 0.16}{3} = 0.07.$$

Средняя BCE (натуральный логарифм) равна

$$-\frac{1}{3}[\log(0.9) + \log(0.8) + \log(0.6)] \approx 0.2798.$$

Числа 0.07 и 0.2798 нельзя напрямую сравнивать как «лучше/хуже»: это значения *разных функций потерь*.

5. Ручной вывод обучения линейного автоэнкодера

5.1 Простая модель $2 \to 1 \to 2$

Возьмём $x = (x_1, x_2)^\top$, скалярный латентный код $z = w_1 x_1 + w_2 x_2$ и линейный декодировщик $\widehat{x} = (v_1 z, v_2 z)^\top$. Для одного наблюдения

$$E = \frac{1}{2}\big[(\widehat{x}_1 - x_1)^2 + (\widehat{x}_2 - x_2)^2\big]. \tag{8}$$

Обозначим $e_j = \widehat{x}_j - x_j$. Тогда по правилу цепочки

$$\frac{\partial E}{\partial v_j} = e_j z, \qquad \frac{\partial E}{\partial w_i} = \big(e_1 v_1 + e_2 v_2\big) x_i. \tag{9}$$

Получили два уровня передачи ошибки: сначала через декодировщик, затем через кодировщик. Это обычный Backpropagation из лекции 4, применённый к задаче самовосстановления.

5.2 Численный пример одного шага

Пусть $x = (1, 2)^\top$, $w = (0.2, 0.1)$, $v = (0.5, 1)$, скорость обучения $\eta = 0.1$. Тогда $z = 0.2 \cdot 1 + 0.1 \cdot 2 = 0.4$, $\widehat{x} = (0.2, 0.4)$, $e = (-0.8, -1.6)$. Ошибка $E = \frac{1}{2}(0.64 + 2.56) = 1.6$. Из (9):

$$\begin{aligned} \nabla_v E &= (-0.32, -0.64), \\ e_1 v_1 + e_2 v_2 &= -0.8 \cdot 0.5 - 1.6 \cdot 1 = -2, \\ \nabla_w E &= (-2, -4). \end{aligned}$$

Обновлённые параметры составляют $v' = (0.532, 1.064)$ и $w' = (0.4, 0.5)$. При новом прямом проходе $z' = 1.4$, $\widehat{x}' = (0.7448, 1.4896)$, поэтому

$$E' = \frac{1}{2}\big[(0.7448 - 1)^2 + (1.4896 - 2)^2\big] \approx 0.1628 < 1.6.$$

Полученное уменьшение ошибки характерно для этого примера, но *один шаг градиентного спуска не обязан уменьшать функцию потерь* при произвольной скорости обучения.

5.3 Матричная форма градиента

Для $z = Wx$, $\widehat{x} = Vz$ и $E = \frac{1}{2}\|\widehat{x} - x\|^2$ имеем

$$\nabla_V E = (\widehat{x} - x) z^\top, \qquad \nabla_W E = V^\top (\widehat{x} - x) x^\top.$$

Размерности: если $W \in \mathbb{R}^{k \times d}$, $V \in \mathbb{R}^{d \times k}$, то $\nabla_W E \in \mathbb{R}^{k \times d}$ и $\nabla_V E \in \mathbb{R}^{d \times k}$. Практически полезно проверять градиенты конечными разностями, когда модель реализуется с нуля.

6. Линейный автоэнкодер и PCA

6.1 Постановка метода главных компонент

Для центрированных наблюдений $x_i \in \mathbb{R}^d$ метод PCA ищет ортонормированный базис $U \in \mathbb{R}^{d \times k}$ ($U^\top U = I_k$), минимизирующий

$$\sum_{i=1}^{N} \|x_i - U U^\top x_i\|_2^2. \tag{10}$$

Столбцы оптимальной матрицы U соответствуют собственным векторам выборочной ковариационной матрицы при наибольших собственных значениях. Если X_c — матрица центрированных данных, разложение $X_c = P \Sigma Q^\top$ даёт направления главных компонент через столбцы Q.

6.2 Связь с линейным AE

Линейный однослойный AE без нелинейностей и при MSE может найти то же *главное подпространство*, что PCA, при подходящих условиях оптимальности. Однако матрицы кодировщика и декодировщика вообще не обязаны совпадать с ортонормированными компонентами PCA: замена $W \mapsto AW$ и $V \mapsto VA^{-1}$ с обратимой A сохраняет отображение VW. Поэтому утверждение «веса линейного AE всегда равны векторам PCA» неверно [8, 6].

> **Когда полезно PCA как baseline** Перед обучением сложного AE следует сравнить качество реконструкции с PCA при одинаковой размерности k. Если глубокая модель не улучшает релевантную метрику либо улучшает незначительно, её вычислительная сложность может быть неоправданной.

6.3 Пример на двумерных данных

Пусть большинство данных примерно удовлетворяет $x_2 \approx 2x_1$. Тогда вектора лежат в узкой полосе около прямой. Одномерный PCA сохраняет координату вдоль этой прямой и отбрасывает небольшие отклонения. Одномерный линейный AE может воспроизводить проекцию на то же подпространство. Если данные лежат на изогнутой кривой, нелинейный AE может моделировать более богатое отображение, но результат зависит от архитектуры и процесса обучения.

7. Варианты автоэнкодеров и регуляризация представлений

7.1 Разреженный автоэнкодер (Sparse AE)

Смысл разреженности: для конкретного объекта активна лишь небольшая часть латентных координат. Один из вариантов функции потерь:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \frac{1}{N} \sum_{i=1}^{N} \|f_\phi(x_i)\|_1. \tag{11}$$

Здесь регуляризуются *активации*, а не обязательно веса. Альтернативой служит KL-штраф на средние активации скрытых нейронов относительно целевого уровня ρ:

$$\mathrm{KL}(\rho \| \widehat{\rho}_j) = \rho \log \frac{\rho}{\widehat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\widehat{\rho}_j}.$$

Слишком сильная регуляризация может обнулить почти все коды и вызвать underfitting. Параметр λ выбирают по валидационным данным, а не по тесту.

7.2 Шумоподавляющий автоэнкодер (Denoising AE)

Пусть $\widetilde{x} \sim q(\widetilde{x} \mid x)$ — испорченный вход, например $\widetilde{x} = x + \varepsilon$, где $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$. Тогда задача

$$\min_{\phi,\theta} \; \mathbb{E}_{x \sim p_{\text{data}},\, \widetilde{x} \sim q(\cdot|x)} \big[\ell\big(x, g_\theta(f_\phi(\widetilde{x}))\big) \big]. \tag{12}$$

Ключ: вход сети испорчен, но целевой выход — чистый объект. Если случайно использовать испорченный вход и в качестве цели, модель не будет специально обучаться восстановлению чистого сигнала. Идея DAE подробно исследована в [9].

7.3 Contractive AE

Контрактивный автоэнкодер штрафует чувствительность латентного отображения к малым изменениям входа:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \big\| J_f(x) \big\|_F^2, \qquad J_f(x) = \frac{\partial f_\phi(x)}{\partial x}. \tag{13}$$

Норма Фробениуса якобиана измеряет локальную чувствительность признаков. Такой штраф может способствовать устойчивости в локальных окрестностях обучающих данных, но вычисление якобиана на больших сетях дорого. Не следует путать его с обычным Weight Decay: это ограничения разных величин.

7.4 Свёрточный автоэнкодер (Convolutional AE)

Для изображений использование полносвязного входа огромной размерности приводит к большому числу параметров. CNN-кодировщик использует локальные фильтры и downsampling, а декодировщик — интерполяцию с последующей свёрткой или транспонированные свёртки. Например, для входа $1 \times 28 \times 28$ можно использовать

$$(1, 28, 28) \to (16, 14, 14) \to (32, 7, 7) \to z \in \mathbb{R}^{16} \to (32, 7, 7) \to (16, 14, 14) \to (1, 28, 28).$$

Размерности необходимо проверять формулами из лекции 7. Транспонированная свёртка не является математическим обращением обычной свёртки в общем случае; при неверной настройке она создаёт «шахматные» артефакты. Альтернатива — upsampling с обычной свёрткой.

7.5 Пример для временных рядов

Для временного окна $x = (x_{t-m+1}, \ldots, x_t)$ последовательность можно закодировать с помощью GRU/LSTM, а декодировать в восстановленное окно. Однако для детектора аномалий временного ряда следует учитывать причинность, сезонность и нестационарность. Кодировщик, использующий будущие точки, недопустим при задаче прогноза в реальном времени; при ретроспективном анализе это может быть допустимо, если задача определена соответствующим образом.

8. Обнаружение аномалий по ошибке реконструкции

8.1 Идея и ограничения

Обучаем автоэнкодер преимущественно на нормальных объектах. Предполагается, что типичные объекты модель восстанавливает лучше, чем нетипичные. Для объекта x определим аномальный балл

$$s(x) = \frac{1}{d} \sum_{j=1}^{d} \left(x_j - \widehat{x}_j\right)^2. \tag{14}$$

Правило решения: $\widehat{y}(x) = \mathbf{1}\{s(x) > \tau\}$, где τ — порог, подобранный заранее. Но мощный AE иногда хорошо восстанавливает и аномалии; в других случаях нормальные, но редкие объекты дают высокую ошибку. Поэтому реконструкция — лишь *признак нетипичности*, а не доказательство аномальности.

8.2 Честное разбиение данных

Для детектора в производственной среде полезно разделить наблюдения на:

1. **train**: нормальные примеры для настройки весов;
2. **validation/calibration**: нормальные данные для порога или, при наличии разметки, выбор порога по заранее заданной метрике;
3. **test**: независимые нормальные и аномальные наблюдения для финальной оценки.

Если данные временные или принадлежат одному устройству/пациенту, разбиение должно предотвращать утечку информации между группами и во времени. Нормализацию и выбор признаков подгоняют *только на train*.

8.3 Порог по квантили

При отсутствии размеченных аномалий можно взять τ равным q-квантили баллов нормальной калибровочной выборки, например $q = 0.99$. Тогда при репрезентативности калибровочного распределения ожидаемая доля нормальных наблюдений выше порога около одного процента. Это не универсальная гарантия доли ложных тревог при сдвиге распределения или сильной зависимости примеров.

8.4 Численный пример порога и решения

Пусть нормальные калибровочные ошибки равны $0.002, 0.004, 0.007, 0.009, 0.012$. При условном выборе порога $\tau = 0.010$ объекты с баллами $0.003, 0.008, 0.025$ классифицируются как normal, normal, anomaly. Если же взять максимум калибровочных ошибок 0.012, классификация первых двух не изменится. Для реальной задачи оценку порога проводят на достаточно большой выборке и документируют правило квантиля, чтобы результат можно было воспроизвести.

8.5 Метрики и зависимость от дисбаланса

Пусть TP — найденные аномалии, FP — нормальные объекты, ошибочно помеченные аномальными, FN — пропущенные аномалии. Тогда

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2PR}{P + R}.$$

При сильном дисбалансе Accuracy способна вводить в заблуждение. Дополнительно используют PR-AUC и ROC-AUC, но PR-AUC особенно информативна, когда редкий класс аномалий является положительным. Необходимо оценивать также среднее число ложных тревог за сутки и стоимость пропуска аномалии.

8.6 Почему нельзя выбирать порог по тестовой выборке

Подбор τ ради максимума F_1 на test делает test частью обучения модели, даже если веса нейросети уже фиксированы. Такая процедура приводит к завышенной оценке. При наличии меток аномалий для настройки порога используется отдельная валидационная выборка; тестовые метрики сообщают после фиксации всех параметров.

9. Переход к генеративным моделям: мотивация VAE

9.1 Почему обычный AE не гарантирует хорошей генерации

Кодировщик детерминированного AE отображает x в точку $z = f_\phi(x)$. Латентные коды могут занимать отдельные области пространства, разделённые пустотами. Если взять случайную точку $z \sim \mathcal{N}(0, I)$, она может попасть в область, где декодировщик никогда не обучался: результат не обязан походить на реальные данные. Для осмысленной генерации требуется контролировать распределение латентных переменных.

9.2 Вероятностный взгляд

В VAE предполагается латентная переменная z и модель совместного распределения

$$p_\theta(x, z) = p(z)\, p_\theta(x \mid z). \tag{15}$$

Здесь $p(z)$ — *prior*, часто стандартное нормальное $\mathcal{N}(0, I)$; $p_\theta(x \mid z)$ — распределение наблюдения при данном коде, определяемое декодировщиком. Генерация происходит в два этапа: сначала $z \sim p(z)$, затем $x \sim p_\theta(x \mid z)$.

9.3 Почему нельзя просто вычислить точный апостериорный закон

Точный апостериорный закон

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)p(z)}{p_\theta(x)}, \qquad p_\theta(x) = \int p_\theta(x \mid z)p(z)\, dz \tag{16}$$

обычно сложен для вычисления из-за многомерного интеграла. Поэтому вводят параметрическое приближение $q_\phi(z \mid x)$ — *вариационный кодировщик*. Его параметры — функции $\mu_\phi(x)$ и $\sigma_\phi(x)$, выдаваемые нейросетью.

10. Основные формулы вариационного автоэнкодера

10.1 Параметризация гауссовского кодировщика

Стандартный выбор:

$$q_\phi(z \mid x) = \mathcal{N}\big(z; \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))\big), \quad p(z) = \mathcal{N}(0, I). \tag{17}$$

Кодировщик возвращает не непосредственно z, а векторы $\mu \in \mathbb{R}^k$ и $\log \sigma^2 \in \mathbb{R}^k$. Использование логарифма дисперсии удобно: сама дисперсия вычисляется как экспонента и всегда положительна. Для численной устойчивости диапазон лог-дисперсии иногда ограничивают, понимая, что это меняет параметризацию модели.

10.2 KL-дивергенция

Мера расхождения между распределениями q и p определяется как

$$D_{\mathrm{KL}}(q\|p) = \mathbb{E}_{z\sim q}\left[\log \frac{q(z)}{p(z)}\right] \geq 0. \tag{18}$$

KL обычно *несимметрична*: $D_{\mathrm{KL}}(q\|p)$ не равно $D_{\mathrm{KL}}(p\|q)$. Если $q = p$ почти всюду, она равна нулю. Строго говоря, при отсутствии абсолютной непрерывности KL может быть бесконечной.

11. Математический вывод ELBO шаг за шагом

11.1 Цель максимизации правдоподобия

Мы хотим максимизировать $\log p_\theta(x)$. Для любого $q_\phi(z \mid x)$, где выражения определены,

$$\log p_\theta(x) = \mathbb{E}_{q_\phi}\left[\log p_\theta(x)\right] \tag{19}$$

$$= \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right] + \mathbb{E}_{q_\phi}\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right] \tag{20}$$

$$= \underbrace{\mathbb{E}_{q_\phi}[\log p_\theta(x,z) - \log q_\phi(z \mid x)]}_{\mathrm{ELBO}(x)} + D_{\mathrm{KL}}\big(q_\phi(z \mid x)\|p_\theta(z \mid x)\big). \tag{21}$$

Поскольку KL неотрицательна,

$$\boxed{\mathrm{ELBO}(x) \leq \log p_\theta(x).} \tag{22}$$

Это и есть *Evidence Lower Bound*. Равенство достигается, если вариационный апостериорный закон совпадает с истинным в смысле нулевой KL.

11.2 Разложение на реконструкцию и KL

Поскольку $p_\theta(x, z) = p_\theta(x \mid z)p(z)$, получаем

$$\begin{aligned} \mathrm{ELBO}(x) &= \mathbb{E}_{q_\phi}\left[\log p_\theta(x \mid z)\right] + \mathbb{E}_{q_\phi}\left[\log p(z) - \log q_\phi(z \mid x)\right] && (23)\\ &= \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]}_{\text{качество реконструкции}} - \underbrace{D_{\mathrm{KL}}(q_\phi(z \mid x)\|p(z))}_{\text{регуляризация латентного закона}}. && (24) \end{aligned}$$

Минимизируемая функция потерь (отрицательная ELBO):

$$\boxed{\mathcal{L}_{\mathrm{VAE}}(x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] + D_{\mathrm{KL}}(q_\phi(z \mid x)\|p(z)).} \tag{25}$$

Важно отличать *математическое правдоподобие* от произвольно нормированной MSE. При гауссовском декодировщике с фиксированной дисперсией негативное лог-правдоподобие пропорционально сумме квадратов ошибок с определённым коэффициентом и добавочной константой; при Bernoulli-декодировщике получается BCE по пикселям.

11.3 Вывод KL для диагональных гауссовских распределений

Для $q = \mathcal{N}(\mu, \mathrm{diag}(\sigma_1^2, \ldots, \sigma_k^2))$ и $p = \mathcal{N}(0, I)$ используется формула

$$\boxed{D_{\mathrm{KL}}(q\|p) = \frac{1}{2}\sum_{j=1}^{k}\left(\mu_j^2 + \sigma_j^2 - 1 - \log\sigma_j^2\right).} \tag{26}$$

Чтобы понять её, напомним логарифм плотности k-мерного гауссовского распределения:

$$\log q(z) = -\frac{k}{2}\log(2\pi) - \frac{1}{2}\sum_j \log\sigma_j^2 - \frac{1}{2}\sum_j \frac{(z_j - \mu_j)^2}{\sigma_j^2}.$$

А для стандартного нормального prior

$$\log p(z) = -\frac{k}{2}\log(2\pi) - \frac{1}{2}\sum_j z_j^2.$$

Берём математическое ожидание разности по q. Учитываем $\mathbb{E}_q[(z_j-\mu_j)^2] = \sigma_j^2$ и $\mathbb{E}_q[z_j^2] = \mu_j^2 + \sigma_j^2$; подстановка даёт (26).

11.4 Численный расчёт KL

Пусть $k = 2$, $\mu = (0.5, -0.2)$, $\sigma^2 = (1, 0.25)$. Тогда

$$\begin{aligned} D_{\mathrm{KL}} &= \frac{1}{2}\left[(0.25 + 1 - 1 - \log 1) + (0.04 + 0.25 - 1 - \log 0.25)\right] \\ &= \frac{1}{2}[0.25 - 0.71 + 1.386294] \approx 0.463147. \end{aligned}$$

Если обе средние равны нулю и обе дисперсии единичны, KL равна нулю. Если μ далека от нуля или дисперсии отличаются от единицы, KL возрастает.

12. Трюк репараметризации и дифференцируемое обучение

12.1 Проблема прямого сэмплирования

Пусть мы случайно генерируем $z \sim q_\phi(z \mid x)$. Для Backpropagation требуется вычислять производную ожидаемой реконструкционной потери по ϕ. Само по себе абстрактное сэмплирование из распределения, параметры которого зависят от ϕ, неудобно для прямой автоматической дифференциации вычислительного графа.

12.2 Замена случайной величины

Положим $\varepsilon \sim \mathcal{N}(0, I)$ независимо от параметров, а затем

$$\boxed{z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon.} \tag{27}$$

Тогда z имеет нужное нормальное распределение, но случайность находится во внешнем ε, а путь от ϕ к z дифференцируем. Это *pathwise gradient estimator*, лежащий в основе VAE [10, 11].

12.3 Проверка производных

Для одного латентного измерения $z = \mu + e^{\ell/2}\varepsilon$, где $\ell = \log \sigma^2$, получаем

$$\frac{\partial z}{\partial \mu} = 1, \qquad \frac{\partial z}{\partial \ell} = \frac{1}{2} e^{\ell/2}\varepsilon.$$

Пусть $\ell = \log 0.25$, $\mu = 0.5$, $\varepsilon = -1$: тогда $\sigma = 0.5$, $z = 0$, $\partial z/\partial \ell = -0.25$. Эти величины может передать автодифференцирование PyTorch.

Принципиальная оговорка Репараметризация не устраняет стохастичность обучения: при конечном числе сэмплов остаётся дисперсия оценки градиента. Она лишь переводит зависимость от параметров в дифференцируемое преобразование случайной величины с фиксированным распределением.

12.4 Алгоритм одного обновления VAE

1. Получить пакет $x_1, \dots, x_B$.
2. Кодировщик вычисляет μ_i и $\ell_i = \log \sigma_i^2$.
3. Сэмплировать независимые $\varepsilon_i \sim \mathcal{N}(0, I)$.
4. Вычислить $z_i = \mu_i + \exp(\ell_i/2) \odot \varepsilon_i$.
5. Декодировщик формирует параметры $p_\theta(x_i \mid z_i)$.
6. Рассчитать реконструкционное отрицательное log-likelihood и KL.
7. Усреднить потери по объектам пакета, вызвать Backpropagation и обновить веса.

13. Выбор вероятностной модели декодировщика

13.1 Bernoulli likelihood

Если пиксель бинарный $x_j \in \{0, 1\}$ и декодировщик возвращает логит a_j, то $\pi_j = \sigma(a_j)$ задаёт вероятность единицы. Отрицательное лог-правдоподобие по координатам совпадает с суммой BCE. На MNIST часто используют нормированные изображения $[0, 1]$ и BCE как практический суррогат, однако для строго вероятностной интерпретации следует уточнять природу наблюдаемой величины.

13.2 Gaussian likelihood

Для непрерывного $x \in \mathbb{R}^d$ модель

$$p_\theta(x \mid z) = \mathcal{N}\big(x; m_\theta(z), s^2 I\big)$$

даёт

$$-\log p_\theta(x \mid z) = \frac{d}{2} \log(2\pi s^2) + \frac{1}{2s^2} \|x - m_\theta(z)\|_2^2.$$

Если s^2 фиксировано, оптимизация этого слагаемого эквивалентна минимизации MSE с подходящим коэффициентом. Если дисперсия обучается,

требуется контролировать численную устойчивость и корректно рассчитывать нормировочные члены; простая подстановка $\|x - \widehat{x}\|^2$ без коэффициентов уже не является точной ELBO.

13.3 Масштаб реконструкции и KL

Если реконструкцию усреднять *по пикселям*, а KL суммировать *по латентным координатам*, меняется относительный вес слагаемых. Воспроизводимое экспериментальное описание должно фиксировать редукцию: например, сумма BCE по 784 пикселям каждого изображения, сумма KL по k латентным переменным, затем среднее по батчу. Иначе сравнение двух реализаций может быть бессмысленным.

14. Beta-VAE, disentanglement и posterior collapse

14.1 Изменение веса KL

В β-VAE используют

$$\mathcal{L}_\beta = -\mathbb{E}_q[\log p_\theta(x \mid z)] + \beta D_{\mathrm{KL}}(q_\phi(z \mid x) \| p(z)). \tag{28}$$

При $\beta = 1$ это стандартная отрицательная ELBO выбранной модели; при $\beta \neq 1$ цель оптимизации модифицируется. При $\beta > 1$ сильнее стимулируется близость апостериорного приближения к prior, что может приводить к более структурированному представлению, но обычно ухудшает точность реконструкции. Обещать автоматическую «интерпретируемость» всех латентных координат неверно [12, 13].

14.2 Posterior collapse

В некоторых конфигурациях модель может приблизиться к $q_\phi(z \mid x) \approx p(z)$ и игнорировать информацию о x в латентном коде. Это называют *posterior collapse*. Особенно проблема встречается с мощными декодировщиками. Диагностические признаки: KL почти нулевая, реконструкция мало зависит от z. Возможные меры включают KL warm-up, изменение архитектуры и ограничение силы декодировщика; ни один приём не является универсальной гарантией.

14.3 Интерполяция в латентном пространстве

Для двух кодов z_a, z_b рассматривают $z(t) = (1-t)z_a + tz_b$, $0 \le t \le 1$, и декодируют промежуточные точки. Плавные изображения на траектории свидетельствуют о некоторой локальной регулярности декодировщика, но визуально плавная интерполяция сама по себе не доказывает, что модель выучила истинные независимые факторы вариации данных.

15. Практический пример 1: PCA и линейный AE на NumPy

Первый эксперимент специально не требует глубоких фреймворков и помогает увидеть, что кодирование — задача оптимизации.

Пример 1. PCA: восстановление двумерных данных одномерным представлением

```python
import numpy as np
rng = np.random.default_rng(42)
n = 500
t = rng.normal(size=n)
X = np.column_stack([t, 2*t + 0.15*rng.normal(size=n)])
mu = X.mean(axis=0)
Xc = X - mu
# PCA: right singular vectors define principal directions.
U, s, VT = np.linalg.svd(Xc, full_matrices=False)
W = VT[:1].T  # (2,1)
Z = Xc @ W   # (500,1)
Xhat = Z @ W.T + mu
mse = np.mean((X - Xhat)**2)
print('X:', X.shape, 'Z:', Z.shape)
print('PCA MSE:', round(float(mse), 6))
assert W.shape == (2, 1)
assert mse >= 0
```

Обсуждение. Если убрать центрирование, найденное подпространство может отражать смещение данных, а не главные направления изменчивости. При оценке на отложенных данных и среднее μ, и базис PCA надо вычислять исключительно по обучающей части.

16. Практический пример 2: полносвязный AE на PyTorch

Будем использовать MNIST как образовательный пример. Загрузка данных через `torchvision` при первом запуске требует интернета; после кэширования набора сеть не нужна. Модель сжимает 784 признака до 32 и восстанавливает вход. Состояния `train()` и `eval()` важны для слоёв Dropout и BatchNorm; в показанной минимальной модели их нет, но корректная дисциплина эксперимента сохраняется.

Пример 2. Определение автоэнкодера и модели данных

```python
import torch
from torch import nn
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, transforms

torch.manual_seed(42)
train_full = datasets.MNIST('data', train=True, download=True,
                            transform=transforms.ToTensor())
test_set = datasets.MNIST('data', train=False, download=True,
                          transform=transforms.ToTensor())
train_set, val_set = random_split(
    train_full, [54000, 6000],
    generator=torch.Generator().manual_seed(42))
train_loader = DataLoader(train_set, batch_size=256, shuffle=True)
val_loader = DataLoader(val_set, batch_size=512)
test_loader = DataLoader(test_set, batch_size=512)

class Autoencoder(nn.Module):
    def __init__(self, latent_dim=32):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Flatten(), nn.Linear(784, 256), nn.ReLU(),
            nn.Linear(256, latent_dim))
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 256), nn.ReLU(),
            nn.Linear(256, 784), nn.Sigmoid())

    def forward(self, x):
        z = self.encoder(x)
        return self.decoder(z).view(-1, 1, 28, 28)

model = Autoencoder(latent_dim=32)
print('Parameters:', sum(p.numel() for p in model.parameters()))
```

Ожидаемый вывод

```
Parameters: 419120
```

Анализ архитектуры. Выход декодировщика ограничен интервалом $(0, 1)$ благодаря sigmoid. Для непрерывных пикселей мы используем MSE. Если выбрать `BCEWithLogitsLoss`, необходимо удалить последний sigmoid и подавать логиты непосредственно в функцию потерь.

Пример 3. Полный цикл обучения, валидации и тестирования AE

```python
import copy
import torch.nn.functional as F

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

def evaluate_ae(loader):
    model.eval()
    total, count = 0.0, 0
    with torch.no_grad():
        for x, _ in loader:
            x = x.to(device)
            loss = F.mse_loss(model(x), x, reduction='mean')
            total += loss.item() * x.size(0)
            count += x.size(0)
    return total / count

best_val, best_state = float('inf'), None
for epoch in range(1, 11):
    model.train()
    total, count = 0.0, 0
    for x, _ in train_loader:
        x = x.to(device)
        optimizer.zero_grad(set_to_none=True)
        recon = model(x)
        loss = F.mse_loss(recon, x)
        loss.backward()
        optimizer.step()
        total += loss.item() * x.size(0)
        count += x.size(0)
    train_mse = total/count
    val_mse = evaluate_ae(val_loader)
    print(f'{epoch:02d}: train={train_mse:.5f} val={val_mse:.5f}')
    if val_mse < best_val:
        best_val = val_mse
        best_state = copy.deepcopy(model.state_dict())

model.load_state_dict(best_state)
print('Final test MSE:', evaluate_ae(test_loader))
```

Замечание к воспроизводимости. Конкретные численные значения зависят от устройства, версии библиотек, инициализации, порядка батчей и настроек детерминизма. Здесь намеренно не приведён вымышленный «типичный» результат: студент должен сообщить фактически полученные train, validation и test значения после запуска.

16.1 Визуализация реконструкций

Пример 4. Сравнение исходных и восстановленных изображений

```python
import matplotlib.pyplot as plt
model.eval()
x, y = next(iter(test_loader))
with torch.no_grad():
    rec = model(x[:8].to(device)).cpu()
fig, ax = plt.subplots(2, 8, figsize=(12, 3.3))
for j in range(8):
    ax[0, j].imshow(x[j, 0], cmap='gray', vmin=0, vmax=1)
    ax[1, j].imshow(rec[j, 0], cmap='gray', vmin=0, vmax=1)
    ax[0, j].axis('off')
    ax[1, j].axis('off')
ax[0, 0].set_title('Original')
ax[1, 0].set_title('Reconstruction')
plt.tight_layout()
plt.show()
```

В отчёте необходимо сравнивать примеры разных классов и явно показывать неудачные реконструкции, а не только наиболее красивые.

17. Практический пример 3: Denoising AE

Возьмём предыдущий автоэнкодер и обучим восстановлению исходного изображения после добавления гауссовского шума. Шум вводится *только во вход*, но не в целевой объект. Ограничение `clamp(0,1)` используется для сохранения диапазона изображений и изменяет фактическую модель шума около границ.

Пример 5. Минимальный тренировочный шаг Denoising AE

```
sigma = 0.30
model.train()
x, _ = next(iter(train_loader))
x = x.to(device)
noisy = (x + sigma*torch.randn_like(x)).clamp(0, 1)
optimizer.zero_grad(set_to_none=True)
recon_clean = model(noisy)
loss = F.mse_loss(recon_clean, x)  # clean target!
loss.backward()
optimizer.step()
print('batch shape:', tuple(x.shape))
```

Эксперимент. Для честного сравнения обучите DAE с нуля тем же числом эпох и одинаковыми условиями, затем проверьте ошибки восстановления при $\sigma = 0$, 0.1, 0.2 и 0.3. Если обучить только на одном уровне шума, нельзя автоматически ожидать оптимального качества на всех остальных уровнях.

18. Практический пример 4: детектор аномалий

Следующий пример демонстрирует *методику*, а не готовую медицинскую или финансовую систему. Допустим, мы обучили AE на изображениях одной нормальной категории. В калибровочной части используем только нормальные объекты и определяем 99-й процентиль ошибок. На независимом тесте оцениваем Precision, Recall, F1 и PR-AUC с известной разметкой.

Пример 6. Аномальный балл и порог на калибровочной выборке

```
import numpy as np
from sklearn.metrics import (precision_score, recall_score,
                             f1_score, average_precision_score)

# Replace arrays by actual outputs of a trained model.
# Scores must be computed for individual observations.
```

Пример 6 (продолжение)

```
calibration_scores = np.array([.008, .009, .012, .007,
                               .011, .015, .010, .006])
threshold = float(np.quantile(calibration_scores, 0.99))
test_scores = np.array([.009, .021, .014, .035, .005])
y_test = np.array([0, 1, 0, 1, 0])
pred = (test_scores > threshold).astype(int)
print('threshold:', round(threshold, 5))
print('predictions:', pred.tolist())
print('F1:', f1_score(y_test, pred))
print('AP:', round(average_precision_score(y_test, test_scores), 4))
```

Методическая оговорка. Синтетический маленький массив служит только для проверки интерфейса метрик. Он не позволяет сделать выводов о качестве системы. В реальной лабораторной работе нужно явно описать, какая именно категория принята за нормальную, как формируется выборка и почему тест независим.

19. Практический пример 5: полноценный VAE на PyTorch

Реализуем вероятностный кодировщик, выдающий μ и $\log \sigma^2$, и декодировщик, выдающий *логиты* Bernoulli-наблюдения. Зафиксируем соглашение: реконструкционная BCE и KL суммируются по координатам каждого объекта, затем усредняются по батчу. Для изображения 28×28 такое масштабирование соответствует (25).

Пример 7. Модель VAE и трюк репараметризации

```
class VAE(nn.Module):
    def __init__(self, latent_dim=16):
        super().__init__()
        self.backbone = nn.Sequential(
            nn.Flatten(), nn.Linear(784, 256), nn.ReLU())
        self.fc_mu = nn.Linear(256, latent_dim)
        self.fc_logvar = nn.Linear(256, latent_dim)
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 256), nn.ReLU(),
            nn.Linear(256, 784))  # logits, no sigmoid

    def encode(self, x):
        h = self.backbone(x)
        return self.fc_mu(h), self.fc_logvar(h)

    def reparameterize(self, mu, logvar):
        eps = torch.randn_like(mu)
        return mu + torch.exp(0.5*logvar)*eps
```

Пример 7 (продолжение)

```
    def decode(self, z):
        return self.decoder(z).view(-1, 1, 28, 28)

    def forward(self, x):
        mu, logvar = self.encode(x)
        z = self.reparameterize(mu, logvar)
        return self.decode(z), mu, logvar

def vae_loss(logits, x, mu, logvar, beta=1.0):
    rec = F.binary_cross_entropy_with_logits(
        logits, x, reduction='none')
    rec = rec.flatten(1).sum(1).mean()
    kl = (-0.5*(1+logvar-mu.pow(2)-logvar.exp()))
    kl = kl.sum(1).mean()
    return rec + beta*kl, rec, kl
```

Внимание: `torch.randn_like(mu)` подчиняется глобальному состоянию генератора случайных чисел. При вычислении валидации можно использовать фиксированное число сэмплов или средний код μ для воспроизводимой визуализации, но это две разные процедуры оценки. Для отчёта необходимо указать выбранную методику.

Пример 8. Обучение VAE и раздельное логирование ELBO

```
vae = VAE(latent_dim=16).to(device)
opt = torch.optim.Adam(vae.parameters(), lr=1e-3)

for epoch in range(1, 11):
    vae.train()
    total, total_rec, total_kl, count = 0., 0., 0., 0
    for x, _ in train_loader:
        x = x.to(device)
        opt.zero_grad(set_to_none=True)
        logits, mu, logvar = vae(x)
        loss, rec, kl = vae_loss(logits, x, mu, logvar)
        loss.backward()
        opt.step()
        b = x.size(0)
        total += loss.item()*b
        total_rec += rec.item()*b
        total_kl += kl.item()*b
        count += b
    print(f'Epoch {epoch:02d} | total={total/count:.2f} '
          f'| reconstruction={total_rec/count:.2f} '
          f'| KL={total_kl/count:.2f}')
```

19.1 Генерация из prior

В отличие от обычного AE, VAE определяет способ сэмплирования z из prior. После обучения:

Пример 9. Генерация 16 новых изображений

```
vae.eval()
with torch.no_grad():
    z = torch.randn(16, 16, device=device)
    logits = vae.decode(z)
    images = torch.sigmoid(logits).cpu()
fig, ax = plt.subplots(4, 4, figsize=(6, 6))
for i in range(16):
    ax.flat[i].imshow(images[i, 0], cmap='gray', vmin=0, vmax=1)
    ax.flat[i].axis('off')
plt.tight_layout()
plt.show()
```

Граница интерпретации. Отображаемые значения sigmoid являются вероятностями единицы в Bernoulli-модели; если нужен буквально бинарный случайный образец, следует дополнительно сэмплировать пиксели из Bernoulli. Визуализация вероятностей обычно выглядит более гладкой, чем бинарный сэмпл.

20. Сравнение AE, DAE и VAE: что считать хорошим результатом

Критерий	AE	Denoising AE	VAE
Латентный код	обычно точка	обычно точка	распределение $q_\phi(z\|x)$
Обучающая цель	реконструкция	чистый объект из шумного	реконструкция и KL
Сэмплирование из prior	не задано	не задано	часть модели
Сильная сторона	простая компрессия	устойчивость к заданному шуму	генеративная модель
Ключевой риск	тождественное копирование	несовпадение типа шума	потеря качества реконструкции, collapse

Таблица 2: Методическое сравнение трёх семейств моделей.

20.1 Дизайн воспроизводимого эксперимента

Сравнение должно учитывать одинаковые train/validation/test разбиения, сопоставимые архитектуры, общее вычислительное время или явно заданный бюджет, несколько случайных инициализаций и согласованные метрики. Если AE оценивается по MSE, а VAE обучается по ELBO, не следует сравнивать значения этих *разных loss* напрямую. Для сравнения реконструкции используйте одну и ту же внешнюю метрику на одном test, например среднюю MSE на пиксель.

20.2 Анализ ошибок и визуальная диагностика

Рекомендуется вывести не менее трёх типов примеров: лучшие реконструкции, медианные, худшие; график распределения ошибок; разницу исходного и восстановленного изображения; визуализацию латентных представлений через PCA или UMAP. Низкоразмерная картинка UMAP зависит от параметров метода и не является математическим доказательством корректности генеративной модели.

20.3 Систематические источники ошибок

Некорректная нормировка данных, неподходящий выходной слой, перепутанные размерности тензоров, регуляризация с неправильным масштабом, утечка test при выборе порога, подбор по тестовой ошибке, неверное использование sigmoid с BCEWithLogits, забытый `eval()`, несогласованность диапазонов изображений и нарушение независимости train/test. Эти ошибки могут маскировать реальное качество модели.

21. Математические дополнения: скорость обучения и ограничения модели

21.1 Оптимизация мини-батчами

Полная эмпирическая функция риска AE

$$J(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(x_i, \widehat{x}_i)$$

заменяется стохастической оценкой на пакете $\mathcal{B}$: $\widehat{J} = |\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} \ell_i$. Для независимого равномерного отбора индексов без систематического перекоса математическое ожидание градиента батча соответствует полному градиенту при обычных условиях дифференцируемости. Но последовательные временные окна могут быть зависимы, и простое случайное перемешивание не решает вопрос правильной оценки на будущих периодах.

21.2 Наличие нескольких локальных минимумов

Нейронный AE оптимизирует нелинейную параметризованную задачу. Нет гарантии, что стандартный Adam находит глобальный минимум, а начальная инициализация влияет на решение. Уменьшение train loss при неизменной или растущей validation loss указывает на возможное переобучение; здесь могут помочь уменьшение модели, Early Stopping, Weight Decay или увеличение разнообразия данных.

21.3 AE как преобразователь и как модель вероятности

Детерминированный AE, минимизирующий MSE, *не задаёт автоматически нормированную вероятность* $p(x)$. Поэтому выражение «вероятность объекта по автоэнкодеру» без дополнительной вероятностной модели некорректно. Аналогично у VAE нижняя оценка ELBO не обязана совпадать с точным log-likelihood, а качество сгенерированных изображений не выводится напрямую только из значения ELBO.

22. Развёрнутый учебный кейс: мониторинг датчиков

Рассмотрим инженерную систему со 12 числовыми датчиками, измеряющими температуру, давление, вибрацию и потребление энергии. Для каждого пятиминутного окна формируется вектор статистик: среднее и стандартное отклонение каждого сигнала; получаем $d = 24$ признака. Цель — выявлять окна, необычные по сравнению со штатной работой оборудования.

Этап 1. Выделить последовательность дат и оборудования, относящуюся к нормальной работе. Убедиться, что период обслуживания и ошибки датчиков не попали без контроля в train.

Этап 2. Разделить нормальные наблюдения по времени на train/calibration; тест сформировать из следующих по времени окон, включающих размеченные известные неисправности. Если много оборудования, оценить способность модели переноситься на другие устройства.

Этап 3. Подогнать медианы и масштабы на train, при наличии сильных выбросов использовать робастное масштабирование. Не применять отдельную нормировку по каждой тестовой точке, которая может уничтожать сам аномальный сигнал.

Этап 4. Построить AE $24 \to 16 \to 5 \to 16 \to 24$ с ReLU внутри и линейным выходом. Оптимизировать MSE. Сравнить с PCA, использующим $k = 5$ компонент.

Этап 5. Выбрать порог по независимому калибровочному набору нормальных данных. Провести анализ ошибок на тестовом периоде, рассчитать Recall по известным авариям и количество ложных тревог на 1000 окон. Оценить задержку обнаружения и стабильность при изменении сезонных условий.

Этап 6. Создать Streamlit-интерфейс: загрузка CSV; таблица баллов; график времени; пометка аномальных окон; экспорт отчёта. В документации явно написать, что решение используется для вспомогательного мониторинга и требует операционной проверки.

> **Ожидаемый конечный продукт студенческого проекта** Подготовленный датасет с описанием разбиения; сохранённый файл весов; воспроизводимый код обучения; базовое сравнение с PCA; таблица метрик; графики ошибок; демонстрационная программа; короткий технический отчёт с ограничениями и рекомендациями.

23. Дополнительные задачи с решениями

23.1 Задача 1. Подсчёт параметров

Найти число параметров AE $10 \to 6 \to 2 \to 6 \to 10$ с полносвязными слоями и смещениями. Решение:

$$P = (10 \cdot 6 + 6) + (6 \cdot 2 + 2) + (2 \cdot 6 + 6) + (6 \cdot 10 + 10) = 66 + 14 + 18 + 70 = 168.$$

23.2 Задача 2. Один шаг линейной реконструкции

Для $x = (2, 1)$, $W = (0.5, 0.5)$, $V = (1, 1)^\top$: $z = 1.5$, $\widehat{x} = (1.5, 1.5)$, $E = \frac{1}{2}[(-0.5)^2 + 0.5^2] = 0.25$. Градиент по V равен $(-0.75, 0.75)^\top$; внутренний градиент $V^\top e = -0.5 + 0.5 = 0$, поэтому $\nabla_W E = (0, 0)$ в данной точке. Это показывает, что градиенты разных слоёв могут вести себя по-разному.

23.3 Задача 3. KL для одного измерения

Пусть $q(z|x) = \mathcal{N}(1, 1)$ и $p(z) = \mathcal{N}(0, 1)$. По формуле (26) получаем $D_{\mathrm{KL}} = \frac{1}{2}(1 + 1 - 1 - 0) = 0.5$. Если изменить среднее на ноль при той же дисперсии, штраф исчезает.

23.4 Задача 4. Репараметризация в двух измерениях

Пусть $\mu = (1, -1)$, $\log \sigma^2 = (0, \log 4)$ и $\varepsilon = (0.5, -0.5)$. Тогда $\sigma = (1, 2)$, следовательно, $z = (1.5, -2)$. Из примера видно, что z зависит и от детерминированных параметров кодировщика, и от случайного ε.

23.5 Задача 5. Выбор модели для датчиков

Если датчики дают непрерывные физические величины с разными единицами измерения, некорректно без обоснования применять BCE к сырым значениям температуры и давления. Практическое решение: нормировка по train и MSE/Huber, либо явная вероятностная модель измерительного шума. Для проверки работоспособности необходимо отдельно оценить нетипичные диапазоны каждого датчика.

24. Контрольные вопросы

1. В чём состоит принцип самоконтролируемого обучения автоэнкодера?
2. Почему AE не обязательно нужна разметка классов?
3. Что называют кодировщиком и декодировщиком?
4. Что такое bottleneck и латентное представление?
5. В чём разница между undercomplete и overcomplete AE?
6. Почему MSE не всегда является оптимальной функцией реконструкции?
7. Какие требования предъявляются к BCE и Bernoulli likelihood?
8. Выведите градиенты для линейного AE с MSE.
9. Почему нескольких линейных слоёв недостаточно для нелинейного сжатия?
10. При каких оговорках линейный AE связан с PCA?
11. Почему веса линейного AE не обязаны совпадать с компонентами PCA?
12. Как подсчитать число параметров многослойного AE?
13. Что такое sparse AE и что именно в нём регуляризуется?
14. Как правильно формируется цель при обучении Denoising AE?
15. Какие проблемы решает контрактирующий штраф по якобиану?
16. Какую пользу CNN приносит при кодировании изображений?
17. Почему транспонированная свёртка не обязана быть обратной?
18. Каким образом AE используется для обнаружения аномалий?
19. Почему ошибка восстановления не является универсальной мерой аномальности?
20. Что такое калибровочный набор для порога?
21. Почему тестовую часть нельзя использовать для выбора порога?
22. Какие метрики подходят для редкого положительного класса?
23. Почему обычный AE не обязательно хорошо генерирует новые данные?
24. Сформулируйте факторизацию $p_\theta(x, z)$ в VAE.
25. Чем prior отличается от posterior?
26. Зачем используется аппроксимация $q_\phi(z|x)$?
27. Запишите формулу ELBO и объясните знак неравенства.
28. Выведите разложение ELBO на реконструкцию и KL.
29. Почему KL-дивергенция неотрицательна?
30. Как вычисляется KL для диагональных гауссовских распределений?
31. Почему кодировщик VAE выдаёт μ и $\log \sigma^2$?

32. В чём идея репараметризации $z = \mu + \sigma\varepsilon$?
33. Какого распределения должна быть ε?
34. Чем логиты отличаются от выходов sigmoid?
35. Почему нельзя сочетать sigmoid и BCEWithLogitsLoss без учёта интерфейса?
36. Чем отличается Bernoulli от Gaussian decoder?
37. Как влияет масштабирование BCE и KL на обучение?
38. Что означает параметр β в Beta-VAE?
39. Что такое posterior collapse и как его диагностировать?
40. Что показывает интерполяция между двумя латентными кодами?
41. Почему train MSE недостаточно для сравнения моделей?
42. Как влияет утечка данных при нормировке?
43. Какие графики надо включить в отчёт об AE/VAE?
44. Почему ELBO не равна автоматически точному log-likelihood?
45. В чём главные ограничения AE для технического мониторинга?

25. Задания для самостоятельной работы

1. **Ручной расчёт.** Найдите все градиенты линейного AE $2 \to 1 \to 2$ для нового собственного примера и проверьте их конечными разностями.
2. **Параметры.** Вычислите число параметров сетей $784 \to 128 \to 16 \to 128 \to 784$ и $784 \to 256 \to 64 \to 256 \to 784$.
3. **Сравнение с PCA.** На одном и том же датасете оцените PCA и нелинейный AE при $k = 2, 8, 16, 32$; постройте график «размерность — test MSE».
4. **Шум.** Обучите Denoising AE при $\sigma \in \{0.1, 0.2, 0.3\}$ и определите, как изменяется качество на ранее невиданных уровнях шума.
5. **Разреженность.** Введите L_1-штраф на латентные активации; сравните распределение активаций при нескольких λ.
6. **Аномалии.** Выберите один нормальный класс MNIST и считайте остальные условными аномалиями; опишите ограничения такой постановки, разбиения и метрики PR-AUC.
7. **Математика VAE.** Самостоятельно выведите (26) из логарифмов гауссовских плотностей.
8. **Репараметризация.** Для заданных μ, ℓ и ε вычислите z и две производные вручную; сравните с `torch.autograd`.
9. **VAE.** Обучите VAE с $k \in \{2, 8, 16\}$ и зафиксируйте reconstruction, KL и суммарную loss на train/validation.
10. **Beta-VAE.** Сравните $\beta = 0.5, 1, 2, 4$ при одинаковом бюджете эпох. Объясните компромисс между реконструкцией и KL.
11. **Визуализация.** Сохраните 16 сэмплов из prior, 16 реконструкций и латентные интерполяции; опишите систематические артефакты.
12. **Проектная работа.** Реализуйте демонстрацию обнаружения аномалий с загрузкой CSV и отчётом о выбранном пороге; дополните картой рисков и ограничений.

Форма отчёта: цель, постановка задачи, происхождение данных, схема архитектуры, число параметров, train/validation/test разбиение, функция потерь и её масштаб, гиперпараметры, таблица метрик, графики, пять разобранных ошибок, вывод и ссылки на источники.

26. Краткие итоги и подготовка к следующей лекции

Обычный AE минимизирует ошибку реконструкции и позволяет получить сжатые признаки; разреженные, шумоподавляющие и контрактирующие варианты накладывают специальные ограничения. Для аномалий реконструкционный балл является эвристикой, требующей корректного порога и независимой проверки. VAE формулирует совместную вероятностную модель, максимизирует ELBO и позволяет генерировать данные из prior. Главные математические идеи VAE — KL-дивергенция и репараметризация. В следующей лекции эти знания понадобятся при изучении *механизмов внимания (Attention) и архитектуры Transformer*, где также используются обучаемые представления, но принцип обработки последовательностей принципиально отличается от рекуррентных сетей.

27. Литература

Список литературы

[1] Траск Э. *Грокаем глубокое обучение* (Grokking Deep Learning). Manning, 2019.

[2] Рашид Т. *Создаём нейронную сеть* (Make Your Own Neural Network). 2016.

[3] Géron A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd ed. O'Reilly, 2022.

[4] Chollet F. *Deep Learning with Python*. 2nd ed. Manning, 2021.

[5] Stevens E., Antiga L., Viehmann T. *Deep Learning with PyTorch*. Manning, 2020.

[6] Goodfellow I., Bengio Y., Courville A. *Deep Learning*. MIT Press, 2016. https://www.deeplearningbook.org/

[7] Nielsen M. A. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/

[8] Baldi P., Hornik K. Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima. *Neural Networks* 2(1), 53–58, 1989. https://doi.org/10.1016/0893-6080(89)90014-2

[9] Vincent P., Larochelle H., Bengio Y., Manzagol P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. *ICML*, 2008. https://doi.org/10.1145/1390156.1390294

[10] Kingma D. P., Welling M. Auto-Encoding Variational Bayes. *ICLR*, 2014. https://arxiv.org/abs/1312.6114

[11] Rezende D. J., Mohamed S., Wierstra D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ICML*, 2014. https://arxiv.org/abs/1401.4082

[12] Higgins I. et al. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *ICLR*, 2017. https://openreview.net/forum?id=Sy2fzU9gl

[13] Locatello F. et al. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. *ICML*, 2019. https://arxiv.org/abs/1811.12359

[14] PyTorch Documentation. *torch.nn and torch.optim*. https://docs.pytorch.org/docs/stable/index.html