

Задачи обучения без учителя

Во всех задачах предыдущих параграфов каждому обучающему объекту x_i сопутствовал *правильный ответ* y_i : голос за кандидата, период обращения планеты, рукописная цифра. Такая задача называется обучением **с учителем** (*supervised learning*): учителем выступает разметка, кем-то заранее предоставленная.

На практике разметка часто отсутствует. У онлайн-кинотеатра — терабайты данных о просмотрах и оценках, но никто не объяснил алгоритму, что «Иван — любитель боевиков, а Маша — драм». У поисковой системы — миллиарды веб-страниц, но никто не пометил, какие из них «про физику», а какие «про поэзию». В таких ситуациях говорят об обучении **без учителя** (*unsupervised learning*). Цель — самостоятельно обнаружить структуру в данных, выделить «главные оси», на которых эти данные расположены, и научиться компактно представлять каждый объект.

Этот параграф устроен так. Мы разберём два центральных сюжета — *восстановление матриц по неполным данным* (matrix completion, она же ставшая знаменитой Netflix problem) и *автокодировщики*. Затем покажем, что оба сводятся к универсальной алгебраической конструкции — **сингулярному разложению матриц (SVD)**. В завершении обсудим эмбединги и контрастивное обучение, которые лежат в основе современных систем распознавания лиц, поиска и больших языковых моделей.

Задача восстановления матрицы.

Netflix problem

Историческая справка

В октябре 2006 г. американская компания **Netflix** — тогда ещё сервис проката DVD по почте — объявила публичный конкурс с призом в 1 миллион долларов: первый, кто

построит алгоритм, угадывающий оценки пользователей хотя бы на 10 % точнее, чем собственный алгоритм Netflix, получит всю сумму.

Компания выложила открытый датасет: 480 189 пользователей, 17 770 фильмов и ≈ 100 млн оценок (по 5-балльной шкале). Каждая оценка сопровождалась датой; нужно было предсказывать неизвестные оценки.

Конкурс длился почти три года. Победила команда BellKor's Pragmatic Chaos (объединение исследователей из AT&T, Yahoo, Big Chaos и Pragmatic Theory), и выиграла она 21 сентября 2009 г. с разницей в 20 минут над командой Ensemble. Сердцевиной победившего метода была именно **малоранговая матричная факторизация**, о которой мы сейчас расскажем. Этот сюжет принципиально изменил индустрию рекомендательных систем — сегодня описанные подходы лежат в основе «вам может понравиться» в Кинопоиске, Окко, YouTube, Spotify, Amazon и сотнях других сервисов.

Формальная постановка. Пусть имеется m пользователей и n фильмов. Полная информация об оценках — это матрица $A \in \mathbb{R}^{m \times n}$:

A_{ij} = оценка, которую пользователь i поставил фильму (j).

Однако в реальности *большинство* элементов этой матрицы неизвестны — средний пользователь Netflix посмотрел и оценил лишь несколько десятков фильмов из почти 18 тысяч. Обозначим

$$\Omega \subset \{1, \dots, m\} \times \{1, \dots, n\} \quad (2)$$

множество *наблюдённых* пар (i, j) . Задача восстановления матрицы по неполным данным: по известным A_{ij} , $(i, j) \in \Omega$, восстановить значения A_{ij} для $(i, j) \notin \Omega$ (рис. 3.22).

Задача восстановления матрицы (Netflix problem): $A = UV^T$, ранг $r = 2$

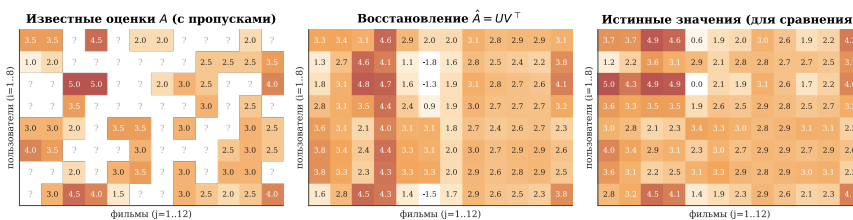


Рис. 3.22. Иллюстрация задачи Netflix на маленьком примере: 8 пользователей, 12 фильмов, ранг $r = 2$. Слева — известные оценки (большинство клеток помечены '?'). В центре — восстановление $\hat{A} = UV^T$ с r -факторизацией (все клетки теперь заполнены, в том числе неизвестные). Справа — истинные значения (для проверки). Метод не просто заполняет пропуски, но и обнаруживает скрытые (латентные) закономерности в данных.

Идея: малоранговая факторизация. Без дополнительных предположений задача безнадежна: восстановить неизвестное число можно как угодно. Но если предположить, что есть *скрытые признаки* — например, «насколько фильм комедийный», «насколько драматичный», «сколько в нём действия» — и описать каждого пользователя i парой чисел, отражающей его пристрастия, а каждый фильм j — такой же парой чисел, то оценка естественным образом получается как «скалярное произведение пристрастий и свойств».

Формализуем. Пусть r — небольшое число (например, $r = 2, 5$ или 20). Введём

- матрицу $U \in \mathbb{R}^{m \times r}$: i -я её строка $\mathbf{u}_i$ — вектор пристрастий пользователя i ;
- матрицу $V \in \mathbb{R}^{n \times r}$: j -я её строка $\mathbf{v}_j$ — вектор свойств фильма j .

Тогда модель оценок:

$$A_{ij} \approx \langle \mathbf{u}_i, \mathbf{v}_j \rangle = \sum_{k=1}^r u_{ik} v_{jk}, \quad \text{или в матричной форме} \quad A \approx UV^T.$$

Сколько параметров в модели? В исходной матрице A — $m \cdot n$ чисел (например, $480\,189 \cdot 17\,770 \approx 8,5$ млрд). В факторизованной модели — $m \cdot r + n \cdot r = (m + n)r$ чисел (для $r = 20$, $m \approx 5 \cdot 10^5$, $n \approx 2 \cdot 10^4$ — около 10^7 , то есть на *три порядка* меньше). Если истинная структура данных — действительно малоранговая, то этой существенно меньшей информации достаточно для восстановления.

i Что значит «ранг»?

Рангом матрицы A называется наименьшее r , при котором A можно представить в виде $A = UV^T$ с матрицами размера $m \times r$ и $n \times r$ соответственно. Эквивалентное определение: ранг r — это максимальное число линейно независимых столбцов (а также строк) матрицы. Чем ниже ранг, тем больше избыточности (повторяющихся закономерностей) в данных — и тем более «сжимаема» матрица.

Постановка как задачи оптимизации. Параметры $\mathbf{u}_i, \mathbf{v}_j$ подбираются так, чтобы модель наилучшим образом соответствовала наблюдаемым данным:

$$\sum_{(i,j) \in \Omega} (A_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j \rangle)^2 \rightarrow \min_{U, V}. \quad (4)$$

Это снова метод наименьших квадратов — но теперь сумма квадратов *только по известным* элементам матрицы. Если интерпретировать данные как «истинные оценки плюс гауссов шум», функционал (3.47) получается из принципа максимума правдоподобия — ровно тем же выводом, что в 3.8.

i Различие задач математической статистики и машинного обучения

Этот пример хорош тем, что наглядно показывает водораздел. В задачах математической статистики (как в 3.1, 3.2) распределение данных *с точностью до неизвестных параметров* известно, и функционал ошибки выводится из принципа максимума правдоподобия.

В задачах машинного обучения и ИИ распределение чаще всего неизвестно; функционал ошибки выбирают исходя из здравого смысла. В нашем примере — квадратичная невязка $(A_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j \rangle)^2$ — это естественный выбор, но он *не выводится* из строгой модели порождения данных. Можно, конечно, постулировать, что $A_{ij} = \langle \mathbf{u}_i, \mathbf{v}_j \rangle + \xi_{ij}$ с $\xi_{ij} \sim \mathcal{N}(0, \sigma^2)$, и тогда квадраты возникают из ОМП — но это уже не вывод из данных, а наше *дополнительное предположение* о структуре шума. Кроме того, в ИИ часто используются нелинейные модели — нейронные сети как универсальные аппроксиматоры.

Неотрицательность. В практической рекомендательной системе оценки лежат в шкале от 1 до 5, а пристрастия и свойства принципиально «качественные»: фильм скорее *в большей или меньшей степени* принадлежит жанру, чем «в антинаправленной». Поэтому естественно ограничиться *неотрицательной* малоранговой факторизацией:

$$A \approx UV^T, \quad U \geq 0, V \geq 0, \quad (5)$$

где неравенство понимается покомпонентно. Такая факторизация называется **неотрицательным матричным разложением** (non-negative matrix factorization, **NMF**); её свойства глубоко изучали Д. Ли и Х. Сын в работах 1999 г.

Пример 3.18. Маленький конкретный пример

Рассмотрим частично заполненную матрицу 4 пользователей и 5 фильмов (оценки от 1 до 5, ‘?’ — неизвестно):

$$A = \begin{pmatrix} 5 & 4 & ? & 1 & ? \\ ? & 5 & 5 & ? & 2 \\ 1 & ? & 2 & 5 & 5 \\ ? & 1 & 1 & 4 & 5 \end{pmatrix}. \quad (6)$$

Видна закономерность: первые два пользователя любят первые три фильма; третий и четвёртый — последние два. Если попытаться найти факторизацию $A \approx UV^T$ с рангом $r = 2$:

$$U = \begin{pmatrix} 1,9 & 0,1 \\ 1,9 & 0,2 \\ 0,1 & 1,7 \\ 0,0 & 1,7 \end{pmatrix}, \quad V = \begin{pmatrix} 2,6 & 0,5 \\ 2,3 & 0,3 \\ 2,5 & 0,5 \\ 0,6 & 2,8 \\ 0,4 & 3,0 \end{pmatrix}, \quad (7)$$

то UV^T почти точно воспроизводит наблюждённые элементы и «предсказывает» неизвестные:

$$UV^T \approx \begin{pmatrix} 5,0 & 4,4 & 4,8 & 1,4 & 1,1 \\ 5,1 & 4,4 & 4,9 & 1,7 & 1,4 \\ 1,1 & 0,7 & 1,1 & 4,8 & 5,1 \\ 0,9 & 0,5 & 0,9 & 4,8 & 5,1 \end{pmatrix}. \quad (8)$$

Обведены восстановленные элементы. Видно: пользователи 1 и 2 в этой модели — любители «жанра 1» (первая координата пристрастий велика); пользователи 3 и 4 — любители «жанра 2». А столбцы V показывают, какому жанру какие фильмы соответствуют.

Алгоритм решения. Задача (3.47) не имеет аналитического решения и относится к нелинейной невыпуклой оптимизации. На практике её решают методом **переменных наименьших квадратов** (Alternating Least Squares, ALS): фиксируем V и оптимизируем U (это уже выпуклая задача и решается по формулам линейной регрессии), потом фиксируем U и оптимизируем V . Чередуем до сходимости. Для очень больших матриц используется стохастический градиентный спуск (3.21).

Три кита и роль малоранговой аппроксимации в ИИ

Разобранный пример хорошо демонстрирует обещание начала параграфа: современный анализ данных стоит на трёх китах.

Вероятностный кит. Хотя ни одного распределения мы явно не вводили, постановка (3.47) имеет естественную вероятностную интерпретацию — «истинные значения плюс гауссов шум». Эта вероятностная подкладка — то, что превращает расплывчатое «найти хорошее восстановление» в строгую задачу.

Оптимизационный кит. Полученная задача оптимизации (3.47) — невыпуклая нелинейная, она требует подходящих численных методов и (для матриц размером в миллиарды элементов) распределённых вычислений на сотнях GPU. Современная индустрия рекомендательных систем — это в первую очередь огромная оптимизационная машинерия.

Линейно-алгебраический кит. Сама постановка — «представить матрицу в виде произведения двух маленьких» — это вопрос линейной алгебры. И тут возникает важная *общая* идея, далеко выходящая за рамки рекомендательных систем.

i LoRA: малоранговые поправки в больших языковых моделях

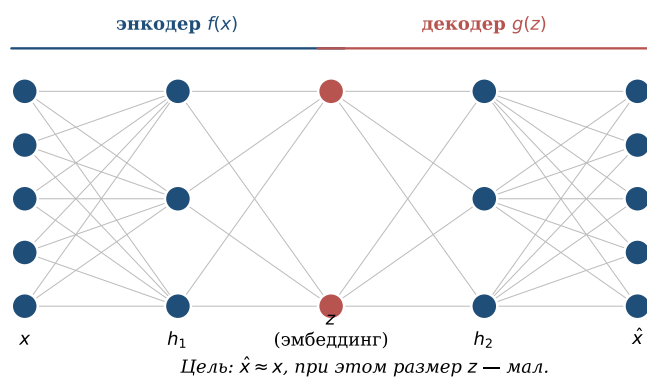
Идея малоранговой аппроксимации играет ключевую роль в современных больших языковых моделях. Когда нужно *дообучить* (fine-tune) готовую модель с миллиардами параметров под узкую задачу (юриспруденция, медицина, диалект), обновлять все веса — слишком дорого. Метод **LoRA** (*Low-Rank Adaptation*, Хью и др., 2021) предлагает: оставить исходные веса W_0 замороженными, а поверх каждой матрицы добавить *малоранговую поправку* $\Delta W = AB^T$, где A, B — матрицы с очень небольшим рангом r (обычно $r = 8$ или 16).

Обновляются только параметры A, B — их в сотни раз меньше, чем параметров полной модели. При этом

результат дообучения почти неотличим от полного. LoRA-адаптеры (один на каждую узкую задачу) весят считанные мегабайты при том, что базовая модель — сотни гигабайтов; их можно «навинчивать» и снимать на лету. Это и доказывает фундаментальную важность идеи малоранговой аппроксимации в современном машинном обучении.

Автокодировщики: общая идея

Второй классический сюжет обучения без учителя — **автокодировщик** (*autoencoder*). Его базовая идея проста и красива: научить нейронную сеть *воспроизводить свой собственный вход*, но сделать это, проходя через узкое «горло» из небольшого числа нейронов. Тогда сеть будет вынуждена сжать всю существенную информацию о входе в этом горле — получится содержательное представление, или **эмбединг** (рис. 3.23).



Архитектура автокодировщика. Энкодер f преобразует вход x в компактный вектор z малой размерности (узкое горлышко). Декoder g пытается из этого z восстановить $\hat{x} \approx x$. Обучение минимизирует $\|x - \hat{x}\|_2^2$ на выборке. Поскольку z — маломе-рен, сеть вынуждена сохранять только самое существенное.

Определение 3.19. Автокодировщик

Автокодировщик — это пара функций $f: \mathbb{R}^n \rightarrow \mathbb{R}^r$ (энкодер) и $g: \mathbb{R}^r \rightarrow \mathbb{R}^n$ (декодер), $r < n$, обученные на выборке $\{x_1, \dots, x_N\} \subset \mathbb{R}^n$ минимизацией функционала

$$\mathcal{L}(f, g) = \frac{1}{N} \sum_{i=1}^N \| \mathbf{x}_i - g(f(\mathbf{x}_i)) \|^2. \quad (9)$$

Вектор $\mathbf{z}_i = f(\mathbf{x}_i) \in \mathbb{R}^r$ называется **эмбеддингом** (*embedding*, скрытым представлением) объекта $\mathbf{x}_i$. Размерность r — *гиперпараметр* модели.

Линейный автокодировщик и связь с PCA

Самый простой случай — когда f и g *линейны*: $f(\mathbf{x}) = W_e \mathbf{x}$, $g(\mathbf{z}) = W_d \mathbf{z}$, где $W_e \in \mathbb{R}^{r \times n}$ и $W_d \in \mathbb{R}^{n \times r}$ — обучаемые матрицы (для простоты считаем, что данные предварительно центрированы, $\bar{\mathbf{x}} = \mathbf{0}$).

Постановка. Функционал (3.49) принимает вид

$$\mathcal{L}(W_d, W_e) = \frac{1}{N} \sum_{i=1}^N \| \mathbf{x}_i - W_d W_e \mathbf{x}_i \|^2 \rightarrow \min_{W_d, W_e}. \quad (10)$$

Решение и его геометрия. Обозначим $X \in \mathbb{R}^{n \times N}$ — матрицу, столбцы которой — центрированные обучающие вектора $\mathbf{x}_1, \dots, \mathbf{x}_N$. Введём *выборочную ковариационную матрицу* $C = \frac{1}{N} X X^T \in \mathbb{R}^{n \times n}$. Эта матрица симметрична и неотрицательно определена.

! Симметричные неотрицательно определённые матрицы

Матрица $C \in \mathbb{R}^{n \times n}$ называется *симметричной*, если $C = C^T$, и *неотрицательно определённой*, если $\langle \mathbf{v}, C \mathbf{v} \rangle \geq 0$ для любого $\mathbf{v} \in \mathbb{R}^n$. *Спектральная теорема* линейной алгебры утверждает: для любой симметричной матрицы C существует ортонормированный базис $\mathbf{e}_1, \dots, \mathbf{e}_n$ из её собственных векторов, $C \mathbf{e}_k = \lambda_k \mathbf{e}_k$. Для неотрицательно определённой матрицы все $\lambda_k \geq 0$. Геометрически: оператор с матрицей C в этом базисе — просто покоординатное растяжение в λ_k раз вдоль k -й оси.

Упорядочим собственные значения C по убыванию: $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$, с соответствующими ортонормированными собственными векторами $\mathbf{e}_1, \dots, \mathbf{e}_n$.

⚠ Теорема 3.20. Решение линейного автокодировщика

Минимум функционала (3.50) (при условии, что f и g линейные, $r < n$) достигается, когда *подпространство*, порождённое строками W_e , совпадает с подпространством, натянутым на первые r собственных векторов $e_1, \dots, e_r$ ковариационной матрицы C . При этом действие энкодера f есть *ортогональная проекция* на это подпространство.

(Доказательство опирается на спектральное разложение и теорему Эккарта–Янга–Мирского, к которой мы перейдём в 3.29.)

Что это означает по содержанию. Линейный автокодировщик *ровно тот же* объект, что **метод главных компонент** (*Principal Component Analysis*, PCA), открытый К. Пирсоном в 1901 г. задолго до всех нейронных сетей.

Линейный автокодировщик с одним нейроном эквивалентен PCA

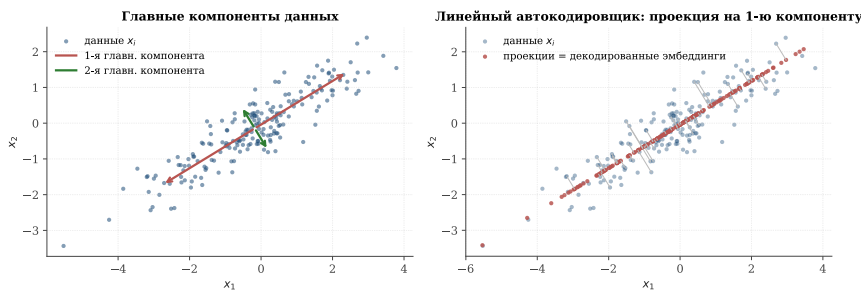


Рис. 3.24. Геометрический смысл линейного автокодировщика. Слева: 2-мерные данные (синие точки) лежат, в основном, вдоль одного направления; красная стрелка — первый собственный вектор ковариационной матрицы (1-я главная компонента); зелёная — второй (2-я главная компонента, существенно короче — дисперсия по этому направлению мала). Справа: линейный автокодировщик с размерностью эмбединга $r = 1$ восстанавливает каждую точку — её ортогональной проекцией на 1-ю главную компоненту (красные точки). Эта проекция и есть «лучшее одномерное представление» исходных двумерных данных в смысле наименьших квадратов.

Геометрически: энкодер — ортогональная проекция в подпространство максимальной дисперсии; декодер — обратное вложение из этого подпространства назад. Эмбединги точек — её координаты в этом подпространстве.

Нелинейные автокодировщики и их разновидности

Принципиальный шаг вперёд — разрешить f и g быть произвольными нейронными сетями. Тогда эмбединг z может «свернуть» в свои несколько координат нелинейные комбинации исходных признаков, что несоизмеримо мощнее линейной проекции.

Свёрточные автокодировщики. Если вход — изображение, естественно использовать свёрточные слои в энкодере (3.20) и обратные операции (deconvolution, или transposed convolution) в декодере. Так строятся *свёрточные автокодировщики*, лежащие в основе ранних систем сжатия изображений и шумоподавления.

Разреженный автокодировщик. Добавим к функционалу (3.49) штрафное слагаемое, требующее, чтобы эмбединг был *разрежён* (т.е. большинство его координат близки к нулю):

$$\mathcal{L}_{\text{sparse}} = \mathcal{L} + \lambda \sum_{i,k} |z_{ik}|, \quad z_{ik} = f(\mathbf{x}_i)_k. \quad (11)$$

Это заставляет сеть «активировать» лишь несколько координат эмбединга для каждого входа — получается интерпретируемое представление, в котором за каждым активным нейроном «закрепляется» свой признак.

Шумоподавляющий автокодировщик. Берём $\mathbf{x}_i$, портим его шумом $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \boldsymbol{\eta}$, а сеть учим восстанавливать *чистое* $\mathbf{x}_i$ по *шумному* $\tilde{\mathbf{x}}_i$:

$$\mathcal{L}_{\text{denoise}} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - g(f(\tilde{\mathbf{x}}_i))\|^2. \quad (12)$$

Это вариант, предложенный П. Венсаном и Й. Бенжио в 2008 г., — очень полезен на этапе предобучения глубоких сетей. Эта идея является концептуальным прародителем современных *диффузионных моделей* (Stable Diffusion, DALL-E — 2022 г. и позднее): они обучаются шаг за шагом удалять гауссов шум из изображения и таким образом порождают совершенно новые картинки из «случайной соли и перца».

Вариационный автокодировщик (VAE). Самая глубокая идейная модификация автокодировщика — **вариационный автокодировщик** (Кингма, Веллинг, 2013). Энкодер выдаёт не сам эмбединг, а параметры *распределения* над эмбедингами:

$$f(\mathbf{x}) = (\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}(\mathbf{x})), \quad (13)$$

после чего сам вектор $\mathbf{z}$ выбирается случайно из нормального распределения $\mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}(\mathbf{x})^2)$ и подаётся в декодер. Обучаемая функция потерь складывается из двух слагаемых:

$$\mathcal{L} = \underbrace{\| \mathbf{x} - g(\mathbf{z}) \|^2}_{\text{невязка восстановления}} + \beta \underbrace{\text{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \| \mathcal{N}(0, 1))}_{\text{регуляризатор}}. \quad (14)$$

Здесь KL — **расхождение Кульбака–Лейблера**, или «расстояние» между двумя вероятностными распределениями (определяется в курсе теории вероятностей). В нашем случае оно *штрафует отклонение* распределения эмбедингов $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ от «эталонного» стандартного нормального $\mathcal{N}(0, 1)$. Содержательно это означает: похожие на вид изображения должны давать близкие $\mathbf{z}$, а всё пространство $\mathbf{z}$ должно быть «равномерно заполненным».

Главное преимущество VAE — возможность *генерации* новых объектов: достаточно выбрать $\mathbf{z}$ случайно из $\mathcal{N}(0, 1)$ и прогнать через декодер; благодаря регуляризатору почти любой такой $\mathbf{z}$ декодируется в осмысленное изображение.

Эмбединги в современных моделях

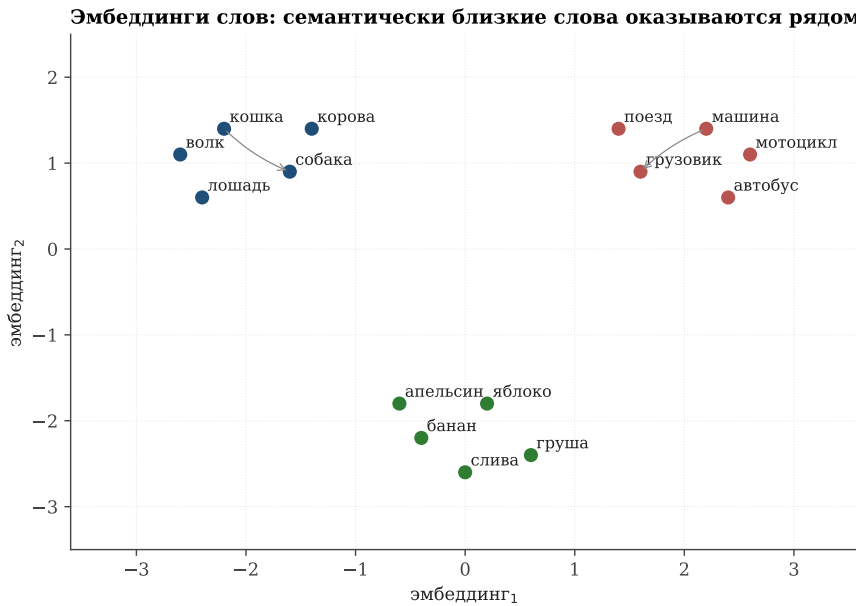
Идея эмбединга вышла далеко за рамки автокодировщиков. Сегодня *любая* глубокая модель ИИ построена вокруг представления своих входов в виде векторов в каком-то многомерном пространстве — их называют общим словом *эмбединги*.

Эмбединги в компьютерном зрении. Рассмотрим обученную свёрточную сеть для классификации изображений (например, ResNet-50 на ImageNet с 1000 классов). Архитектурно её последние два слоя устроены так:

$$\underbrace{\mathbf{x} \rightarrow \dots \rightarrow \mathbf{z} \in \mathbb{R}^{2048}}_{\text{свёрточный «костяк»}} \xrightarrow{W \in \mathbb{R}^{1000 \times 2048}} \mathbf{p} = \text{softmax}(W\mathbf{z}) \in \mathbb{R}^{1000} \quad (15)$$

Вектор $\mathbf{z}$ на предпоследнем слое — это и есть *эмбединг изображения*: сжатое представление в $\mathbb{R}^{2048}$, из которого последнее линейное преобразование W восстанавливает вероятности классов. Эмпирически оказалось, что $\mathbf{z}$ *переносим*: его можно использовать для совсем других задач — поиска похожих фотографий (сравнение по косинусной мере), классификации классов, которых не было в обучении (*transfer learning*), распознавания одного и того же лица на разных снимках. В этом и состоит главная практическая ценность эмбедингов: одна обученная сеть даёт универсальное «представление изображения», пригодное для целого спектра приложений.

Эмбединги в больших языковых моделях. Каждое слово, а точнее — каждый *токен* (подпоследовательность символов), современная LLM кодирует вектором фиксированной длины (обычно 1024–4096 чисел). Эти эмбединги — обучаемые параметры модели; в процессе обучения семантически близкие слова приобретают близкие эмбединги (рис. 3.25).



Двумерная проекция эмбедингов 15 слов на трех семантических кластеров: животные, транспорт, фрукты. Эмбединги получены из обучения модели предсказывать слово по контексту (модель word2vec, 2013). Слова, близкие по смыслу, оказываются близкими в пространстве эмбедингов — хотя никаких меток «это слово относится к классу ‘животные’» модели не давалось! Семантика возникает из статистики.

i Большие языковые модели: краткая анатомия

Современная большая языковая модель (LLM; примеры — GPT-серии от OpenAI, Claude от Anthropic, Llama от Meta, GigaChat от Сбера, YandexGPT от Яндекса) построена на архитектуре **трансформер** (Васвани и др., 2017). На входе модель получает последовательность токенов; каждый токен превращается в эмбединг.

Дальше эти эмбединги многократно прогоняются через слои двух типов:

- *механизм внимания (self-attention)*: каждый токен «смотрит» на другие токены последовательности и обновляет свой эмбединг, учитывая контекст. Это и есть то, что делает модель «контекстно-зависимой»: то же слово в разных контекстах получает разный финальный эмбединг.
- *полносвязный блок (FFN, feedforward network)* — обычный полносвязный слой большой ширины. Именно в этих блоках «хранятся знания» модели — факты, выученные из обучающих текстов.

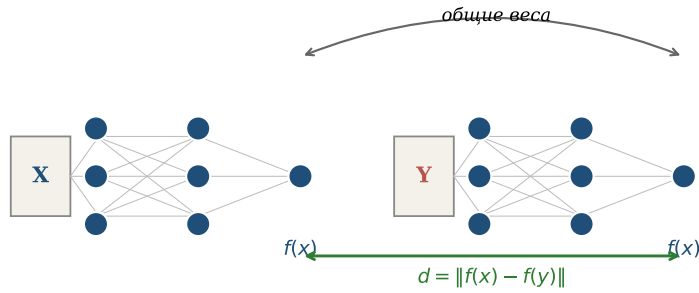
В конце последний эмбединг конвертируется в распределение вероятностей над всеми токенами словаря — и модель предсказывает следующий токен. Так, генерируя по одному токenu за раз, LLM пишет тексты, отвечает на вопросы, сочиняет код — всё это, по сути, последовательное предсказание следующего токена.

Сиамские сети и контрастивное обучение

В автокодировщике мы заставляли сеть восстанавливать вход целиком. Часто это избыточно: для практических задач нам нужно не «восстановить картинку до пикселя», а узнать, *похожи ли две картинки между собой*. Этим занимается **контрастивное (или контрастное) обучение** (*contrastive learning*).

Сиамская сеть. Сиамская сеть (*Siamese network*, Бромли с соавторами, 1993) состоит из двух копий одной и той же нейронной сети f с общими весами. На вход подают *пару* объектов; каждый прогоняется через свою копию f , и в результате получаются два эмбединга $f(x)$ и $f(y)$. Их близость или удалённость характеризует «похожесть» объектов (рис. 3.26).

Сиамская сеть: один и тот же энкодер f применяется к двум входам



Если x, y — похожи, минимизируем d . Если различны — заставляем d быть большим.

Архитектура сиамской сети. Два экземпляра одной и той же нейронной сети f обрабатывают пару объектов x и y . На выходе вычисляется расстояние $d = \|f(x) - f(y)\|$. В процессе обучения параметры f настраиваются так, чтобы d было малым для «похожих» пар (например, двух фотографий одного и того же человека) и большим для «непохожих» (фотографии разных людей). **Контрастивный функционал.** Пусть имеется набор обучающих пар $\{(x_i, y_i, t_i)\}$, где $t_i = 1$, если пара — «положительная» (похожие), $t_i = 0$, если «отрицательная» (непохожие). Простейший *контрастивный* функционал (Хадселл, Шопра, ЛеКун, 2006):

$$\mathcal{L} = \sum_i \left[t_i \cdot d_i^2 + (1 - t_i) \cdot (\max(0, m - d_i))^2 \right], \quad (16)$$

где $d_i = \|f(x_i) - f(y_i)\|$, а $m > 0$ — *отступ (margin)*, гиперпараметр. Смысл: для похожих пар ($t_i = 1$) минимизируется d_i^2 — расстояние «прижимается» к нулю; для непохожих ($t_i = 0$) штрафуются только те, что d_i меньше m — то есть от непохожих пар требуется минимальное расстояние m , а дальше уже всё равно.

Применения.

Распознавание лиц. Сиамская сеть **FaceNet** (Google, 2015) обучается на миллионах фотографий: эмбединг каждого лица — вектор в $\mathbb{R}^{128}$. Положительные пары — два снимка одного человека (часто в разных условиях); отрицательные — разных. После обучения по тестовому снимку можно определить личность простым поиском ближайшего эмбединга в базе. Точность — более 99,6 % на стандартном бенчмарке LFW.

Такая система — основа всех современных систем распознавания лиц.

Поиск изображений по тексту (и наоборот). Модель CLIP (OpenAI, 2021) обучает два энкодера: один — для изображений (свёрточная или трансформерная сеть), другой — для текстов (трансформер). Положительные пары — картинка и её настоящая подпись; отрицательные — картинка и случайная подпись из батча. После обучения эмбединги изображения и эмбединги его осмысленного текстового описания оказываются близкими в общем многомерном пространстве. Это даёт колоссальные возможности: искать картинки по произвольным описаниям («рыжий кот на пианино»), классифицировать изображения в произвольные классы, не имея размеченной выборки (*zero-shot classification*), и многое другое.

Дубликаты в поиске. Тот же контрастивный подход используют поисковые системы (Яндекс, Google) для дедупликации страниц, рекомендательные системы — для поиска похожих товаров, антивирусы — для группировки похожих образцов вредоносного кода.

Сингулярное разложение матриц^{*}

Этот раздел — *со звёздочкой*: материал более продвинутый, и при первом чтении его можно пропустить. Однако именно здесь раскрывается глубокая связь между двумя сюжетами параграфа — матричной факторизацией и линейным автокодировщиком — через одну и ту же алгебраическую конструкцию.

Матрицы как линейные преобразования. Каждая матрица $A \in \mathbb{R}^{m \times n}$ задаёт линейное преобразование $\mathbb{R}^n \rightarrow \mathbb{R}^m$, $x \mapsto Ax$ (см. определение 3.10). Столбцы матрицы A — это образы базисных ортов $e_1, \dots, e_n$: j -й столбец равен Ae_j . Произведение матриц $C = AB$ соответствует суперпозиции преобразований: сначала действует B , потом A .

Простейшие линейные преобразования. Среди всех линейных преобразований $\mathbb{R}^n \rightarrow \mathbb{R}^n$ выделяются два простейших класса.

- **Повороты** — матрицы Q , сохраняющие длины: $\|Qx\| = \|x\|$. Они называются *ортогональными* и характеризуются равенством $Q^T Q = I$.
- **Растяжения по осям** — диагональные матрицы Σ :
 $\Sigma e_k = \sigma_k e_k$, $\sigma_k \geq 0$. Они растягивают k -ю ось в σ_k раз.

Сингулярное разложение. Знаменитый факт линейной алгебры (Бельтрами, 1873; Йордан, 1874; Сильвестр, 1889 для квадратных матриц; Экарт и Янг, 1936 для прямоугольных) — любое линейное преобразование можно представить как комбинацию поворота, растяжения и снова поворота:

Теорема 3.21. Сингулярное разложение (SVD)

Для любой матрицы $A \in \mathbb{R}^{m \times n}$ существует представление

$$A = U \Sigma V^T, \quad (17)$$

где $U \in \mathbb{R}^{m \times m}$ и $V \in \mathbb{R}^{n \times n}$ — ортогональные матрицы ($U^T U = I_m$, $V^T V = I_n$), а $\Sigma \in \mathbb{R}^{m \times n}$ — «диагональная» (т. е. $\Sigma_{ij} = 0$ при $i \neq j$) с неотрицательными элементами на диагонали $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)} \geq 0$. Эти числа называются **сингулярными значениями** матрицы A ; столбцы U и V — **сингулярными векторами**.

Геометрически: преобразование с матрицей A можно представить последовательностью — сначала поворот в исходном пространстве (матрица V^T), затем покоординатное растяжение (матрица Σ), затем поворот в пространстве-образе (матрица U) (рис. 3.27).

Геометрический смысл SVD: $A = U \Sigma V^T$ — поворот, растяжение, поворот

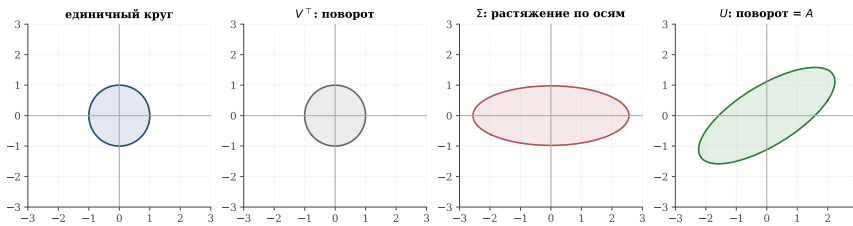


Рис. 3.27. Геометрическая иллюстрация SVD $A = \begin{pmatrix} 0,5 \text{ amp}; 1 \end{pmatrix}$

$U \Sigma V^T$ на примере $\begin{pmatrix} 0,5 \text{ amp}; 1 \end{pmatrix}$. Единичная окружность последовательно: 1) поворачивается матрицей V^T — столбцы v_1, v_2 матрицы V при этом становятся координатными осями; 2) растягивается матрицей Σ по координатным осям — появляется эллипс с полуосями σ_1, σ_2 ; 3) ещё раз поворачивается матрицей U — координатные оси переходят в столбцы u_1, u_2 матрицы U , и итог совпадает с прямым применением A к окружности.

Полезный способ читать SVD: столбцы $v_1, \dots, v_n$ матрицы V — это «входные направления», на которых матрица A действует чище всего (как простое растяжение, без поворота). Столбцы $u_1, \dots, u_m$ матрицы U — соответствующие «выходные направления»: $A v_i = \sigma_i u_i$. Числа σ_i показывают, во сколько

раз растягивается каждое такое направление; по соглашению они упорядочены по убыванию.

Связь с симметричной задачей. Доказательство SVD сводится к спектральной теореме для симметричной матрицы. Заметим:

- $A^T A = (U \Sigma V^T)^T (U \Sigma V^T) = V \Sigma^T U^T U \Sigma V^T = V \Sigma^T \Sigma V^T$ — это спектральное разложение симметричной неотрицательной матрицы $A^T A$. Её собственные значения — σ_k^2 ; собственные векторы — столбцы V .
- Аналогично, $AA^T = U \Sigma \Sigma^T U^T$. Собственные векторы AA^T — столбцы U .

Так одно общее утверждение для произвольных матриц редуцируется к симметричному случаю.

Теорема о наилучшей малоранговой аппроксимации. SVD замечательно тем, что позволяет получить наилучшее приближение матрицы матрицей фиксированного ранга.

⚠ Теорема 3.22. Эккарта–Янга–Мирского (1936)

Пусть $A \in \mathbb{R}^{m \times n}$ и $A = U \Sigma V^T$ — её сингулярное разложение. Для любого $r \leq \min(m, n)$ определим *обрезанное разложение*

$$A_r = \sum_{k=1}^r \sigma_k \mathbf{u}_k \mathbf{v}_k^T = U_r \Sigma_r V_r^T, \quad (18)$$

где U_r, V_r — первые r столбцов матриц U, V , а $\Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r)$. Тогда матрица A_r имеет ранг r и реализует наилучшую среди всех матриц ранга $\leq r$ аппроксимацию исходной матрицы:

$$\min_{\text{rank } B \leq r} \|A - B\|_F = \|A - A_r\|_F = \sqrt{\sigma_{r+1}^2 + \dots + \sigma_{\min(m,n)}^2} \quad (19)$$

где $\|\cdot\|_F$ — норма Фробениуса: $\|M\|_F^2 = \sum_{i,j} M_{ij}^2$.

Иными словами, отбрасывая в разложении (3.52) все сингулярные значения, начиная с $(r+1)$ -го, мы получаем оптимальную малоранговую аппроксимацию. Если первые r сингулярных значений велики, а последующие малы, то A_r — почти такая же хорошая, как A , но в $\frac{mn}{(m+n)r}$ раз более компактна по числу параметров (рис. 3.28).

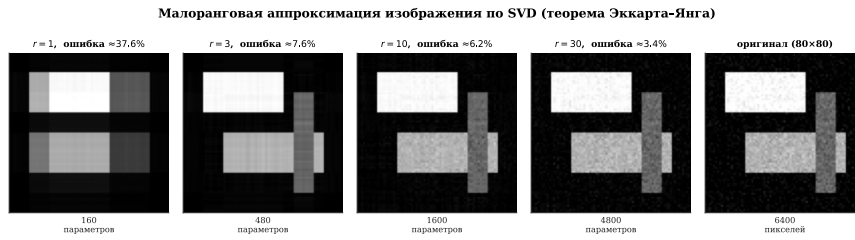


Рис. 3.28. Малоранговая аппроксимация изображения по SVD: восстановление картинки 80×80 при разных рангах. При $r = 1$ — крупная полоса (один «прототип»). При $r = 3$ виден общий силуэт. При $r = 10$ — почти неотличимо от оригинала, но в ~ 4 раза меньше параметров. При $r = 30$ совпадение полное, при том что число параметров на 38 % меньше, чем в оригинальном представлении.

Связь с автокодировщиком и матричной факторизацией.

Линейный автокодировщик из 3.25 — это в точности SVD-факторизация центрированных данных. Если столбцы X — обучающая выборка, и $X = U\Sigma V^T$ её SVD, то оптимальный энкодер — $f(x) = U_r^T x$, оптимальный декодер — $g(z) = U_r z$, где U_r — первые r столбцов матрицы U . Сингулярные значения σ_k показывают, насколько важна k -я главная компонента (большое σ_k — много дисперсии в этом направлении).

Матричная факторизация $A \approx UV^T$ из 3.23 — это (без ограничения на неотрицательность) ровно $A \approx A_r$ из теоремы Эккарта-Янга-Мирского. Если бы все элементы A были известны, идеальной стратегией было бы взять обрезанное SVD-разложение ранга r . В реальной Netflix problem известны лишь $\sim 1\%$ элементов матрицы, и приходится решать задачу (3.47), которая является обобщением SVD на случай пропусков.

Школа Е. Е. Тыртышниковой и тензорные разложения.

Идея малоранговой аппроксимации не ограничивается матрицами. **Тензором** называется многомерный массив чисел: матрица — двумерный тензор, а матрицы с тремя и более индексами — тензоры высших порядков. Тензоры возникают повсюду в современных приложениях: изображение видеоряда — 4-индексный тензор (время $\times$ высота $\times$ ширина $\times$ цвет); веса трансформера — тысячи матриц, организованных в одну сложную структуру; квантово-механические задачи — тензоры с десятками индексов.

Аналог малоранговой аппроксимации для тензоров — значительно более тонкая задача. Современный мощный инструмент — **разложение в виде тензорного поезда** (*tensor train decomposition*), предложенное в 2009 г. российским математиком, профессором РАН **Иваном Валерьевичем Оселедцем**, учеником академика **Евгения Евгеньевича Тыр-**

тышников (МГУ, ИВМ РАН). Тензорный поезд позволяет хранить тензор с d индексами при размере N по каждому индексу с помощью $\sim d N r^2$ параметров вместо N^d — то есть избавляет от знаменитого «проклятия размерности». Этот результат лежит в основе многих современных алгоритмов в квантовой химии, машинном обучении и решении многомерных дифференциальных уравнений.

За эти и связанные работы Е. Е. Тыртышников — один из мировых лидеров в области тензорных и матричных разложений — является лауреатом премий Сбербанка в области искусственного интеллекта.

Резюме параграфа

Обучение без учителя — это семейство задач, в которых модель должна самостоятельно обнаружить структуру в данных без подсказок-меток. В этом параграфе мы разобрали два центральных сюжета.

Восстановление матрицы (Netflix problem):

- Гипотеза малоранговой факторизации $A \approx UV^T$ — из неё естественным образом получается рекомендательная система.
- Функционал ошибки $\sum_{(i,j) \in \Omega} (A_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j \rangle)^2$ — здравый смысл, который при желании можно вывести из ОМП в предположении гауссова шума.
- Идея малоранговой коррекции легла в основу LoRA — стандарта дообучения больших языковых моделей.

Автокодировщики:

- Энкодер f сжимает вход в эмбединг $\mathbf{z}$ малой размерности; декодер g восстанавливает вход из этого эмбединга. Минимизируется $\| \mathbf{x} - g(f(\mathbf{x})) \|^2$.
- В линейном случае оптимальное решение — ортогональная проекция на подпространство максимальной дисперсии (PCA).
- Эмбединги — универсальное представление в современных моделях ИИ: компьютерное зрение, языковые модели, рекомендации.
- Контрастивное обучение через сиамские сети позволяет обучать качественные эмбединги, не требуя восста-

навливать вход целиком — достаточно знать, что одни пары похожи, а другие нет.

Оба сюжета фундаментально связаны через **сингулярное разложение** $A = U\Sigma V^\top$ и теорему Эккарта–Янга–Мирского о наилучшей малоранговой аппроксимации. Это — мощный универсальный инструмент линейной алгебры, лежащий и за рекомендательными системами, и за PCA, и за современными тензорными разложениями (школа Тыртышникова–Оселед-ца).

! Задачи для самостоятельной работы

1. Проверьте, что в примере 3.18 матрицы U и V действительно дают $UV^\top$, близкое к A во всех наблюдаемых клетках.
2. Сколько параметров в модели $A \approx UV^\top$, если $m = 10^6$ пользователей, $n = 10^4$ фильмов, ранг $r = 20$? Во сколько раз это меньше числа элементов исходной матрицы?
3. Покажите, что для центрированной выборки матрица $C = \frac{1}{N}XX^\top$ действительно симметрична и неотрицательно определена.
4. Сделайте свой PCA: возьмите датасет $\mathbf{x}_i \in \mathbb{R}^2$ вашего сочинения (хотя бы 30 точек), посчитайте C , найдите собственные значения и векторы (можно с помощью `numpy.linalg.eig`). Постройте график как на рис. 3.24.
5. Возьмите изображение в градациях серого. Получите его SVD (`numpy.linalg.svd`) и постройте малоранговые аппроксимации с $r = 1, 5, 20, 50$. Постройте график зависимости ошибки в норме Фробениуса от r . Найдите минимальное r , при котором ошибка не превышает 5 % от $\|A\|_F$.
6. * Докажите формулу (3.53), опираясь на SVD и ортогональность U, V в норме Фробениуса.
7. * Сиамская сеть для рукописных цифр. Возьмите MNIST/UCI Digits. Постройте сиамскую сеть, обучающуюся на парах «одна и та же цифра / разные цифры». После обучения покажите, что эмбедин-

ги одной цифры из разных написаний оказываются близкими.