Eigenfaces: как алгоритм узнаёт лицо

Постановка задачи

В 1991 г. Мэттью Тёрк и Алекс Пентланд из MIT поставили простой вопрос: *можно ли обучить компьютер узнавать лицо, не прибегая ни к каким специальным «лицевым» признакам — носу, глазам, контурам, — а просто работая с матрицей пикселей?* Их ответ — метод **собственных лиц** (*eigenfaces*) — стал классикой и открыл целую область computer vision.

Идея: каждая фотография лица — это точка в очень многомерном пространстве. Например, лицо 64×64 — это вектор длины $64 \cdot 64 = 4\,096$ в пространстве $\mathbb{R}^{4096}$. Двух разных людей снимали много раз — получаем облако точек в этом пространстве.

Главное наблюдение: *это облако очень узкое.* Хотя пространство $4\,096$-мерное, реальные лица занимают в нём подпространство размерности всего лишь ~ 50. Найти это подпространство — и есть задача.

Olivetti faces: 40 человек × 10 поз каждого (показано по 1 кадру)

Фрагмент датасета Olivetti Faces: 400 фотографий 64 × 64 пикселя, 40 человек по 10 снимков каждого. Каждая фотография — точка в ℝ4096.

Шаг 1. Среднее лицо и центрирование

Пусть у нас N фотографий $x_1, \ldots, x_N \in \mathbb{R}^d$ (где $d = 4096$ для 64×64). Вычислим **среднее лицо**:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{1}$$

И вычтем его из каждого изображения: $\tilde{x}_i = x_i - \bar{x}$. *Среднее лицо* (рис. 0.5) — это размытая маска, в которой видно «лицо вообще»: усреднённые глаза, нос, рот, овал. Дальше мы будем работать только с отклонениями от этого среднего.

Среднее лицо

Среднее лицо по 400 снимкам датасета Olivetti.

Шаг 2. Главные направления изменчивости

Соберём центрированные изображения в строки матрицы $X \in \mathbb{R}^{N \times d}$ (одна строка = одно лицо). Применим к ней SVD:

$$X = U \, \Sigma \, \bar{V}. \tag{2}$$

Каждая *строка* $^{\top}v_i$ матрицы $\bar{V}$ — это вектор длины $d = 4096$, который можно нарисовать как картинку 64×64. Эти картинки и называются **собственными лицами** — eigenfaces (рис. 0.6). Содержательно:

- v_1 — направление наибольшей изменчивости. Чаще всего это «освещение слева/справа».
- v_2 — направление второй по величине изменчивости (часто «улыбка / не улыбка»).
- $v_3, \ldots$ — постепенно более тонкие признаки: очки, форма щёк, наклон головы, индивидуальные особенности.

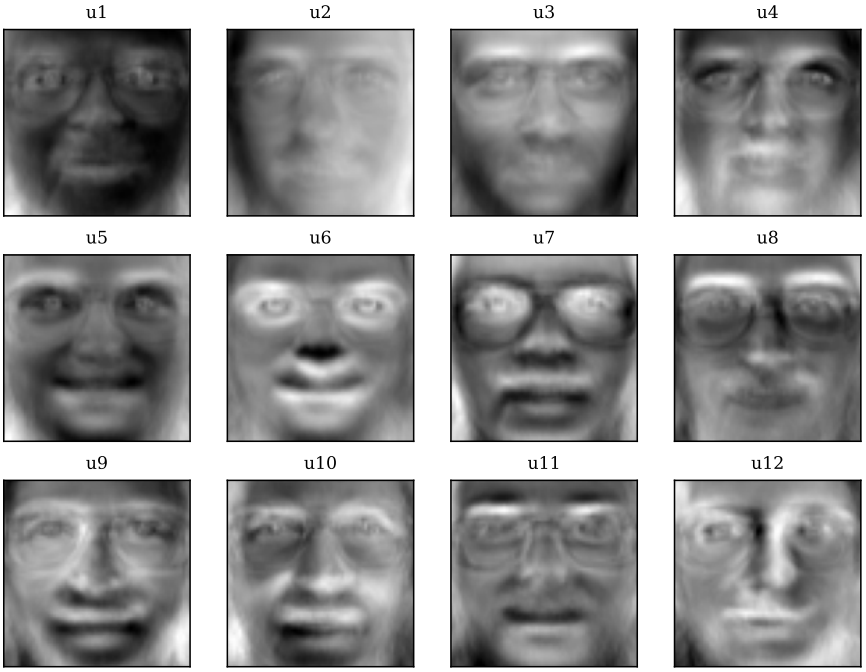

Рис. 0.6. Первые 12 eigenfaces датасета Olivetti, отрисованные как картинки 64 × 64. Эти странные «полу-лица» — координатные оси в пространстве, где живут все настоящие лица.

Шаг 3. Лицо как вектор из 50 чисел

Зафиксируем k (обычно $k = 30 \div 100$) и спроектируем каждое лицо x на эти k собственных лиц:

$$\alpha_i = v_i^\top (x - \bar{x}), \qquad i = 1, \ldots, k. \tag{3}$$

Получили вектор коэффициентов $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{R}^k$. Это и есть «лицо человека» с точки зрения алгоритма — *всего* 50 *чисел* вместо 4096.

Шаг 4. Распознавание и реконструкция

Распознавание. Чтобы узнать, кто на новой фотографии y, вычислим её коэффициенты β по (0.6) и найдём в базе ближайшего соседа: $\hat{\jmath} = \arg\min_j \| \beta - \alpha^{(j)} \|$.

Реконструкция. Зная α, можно восстановить лицо:

$$x \approx \bar{x} + \sum_{i=1}^{k} \alpha_i \, v_i. \tag{4}$$

Рис. 0.7 показывает, как качество реконструкции растёт с k.

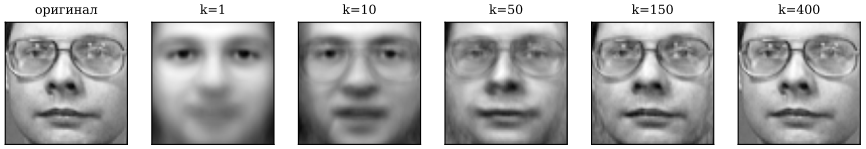

Рис. 0.7. Реконструкция одной из фотографий датасета через первые k eigenfaces. При k = 10 узнаётся «общий тип» лица; при k = 50 — уже конкретный человек; при k = 150 — почти оригинал.

Реализация на Python

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces

# 1. Load: 400 faces, 64x64 each
faces = fetch_olivetti_faces().images       # (400, 64, 64)
X = faces.reshape(400, -1)                  # (400, 4096)

# 2. Center: subtract the mean face
x_bar = X.mean(axis=0)
X_centered = X - x_bar

# 3. SVD => eigenfaces are rows of Vt
U, S, Vt = np.linalg.svd(X_centered, full_matrices=False)

# 4. Project all faces onto top-k eigenfaces
k = 50
alpha = X_centered @ Vt[:k].T               # (400, 50)

# 5. Recognise: nearest neighbour in alpha-space
def recognise(y, k=50):
    y_centered = y.flatten() - x_bar
    beta = Vt[:k] @ y_centered
    distances = np.linalg.norm(alpha - beta, axis=1)
    return distances.argmin()

# 6. Reconstruct a face from k coefficients
def reconstruct(idx, k=50):
    return x_bar + Vt[:k].T @ alpha[idx, :k]
```

Это интересно

Метод eigenfaces — родоначальник всех современных систем распознавания лиц, включая ту, что разблокирует ваш смартфон. Современные системы заменяют SVD на свёрточную нейросеть (см. § 3.3.7), но базовая идея — *сжать лицо в короткий вектор, в котором близкие лица оказываются рядом* — осталась той же. Эти короткие векторы сегодня называют **эмбеддингами**.