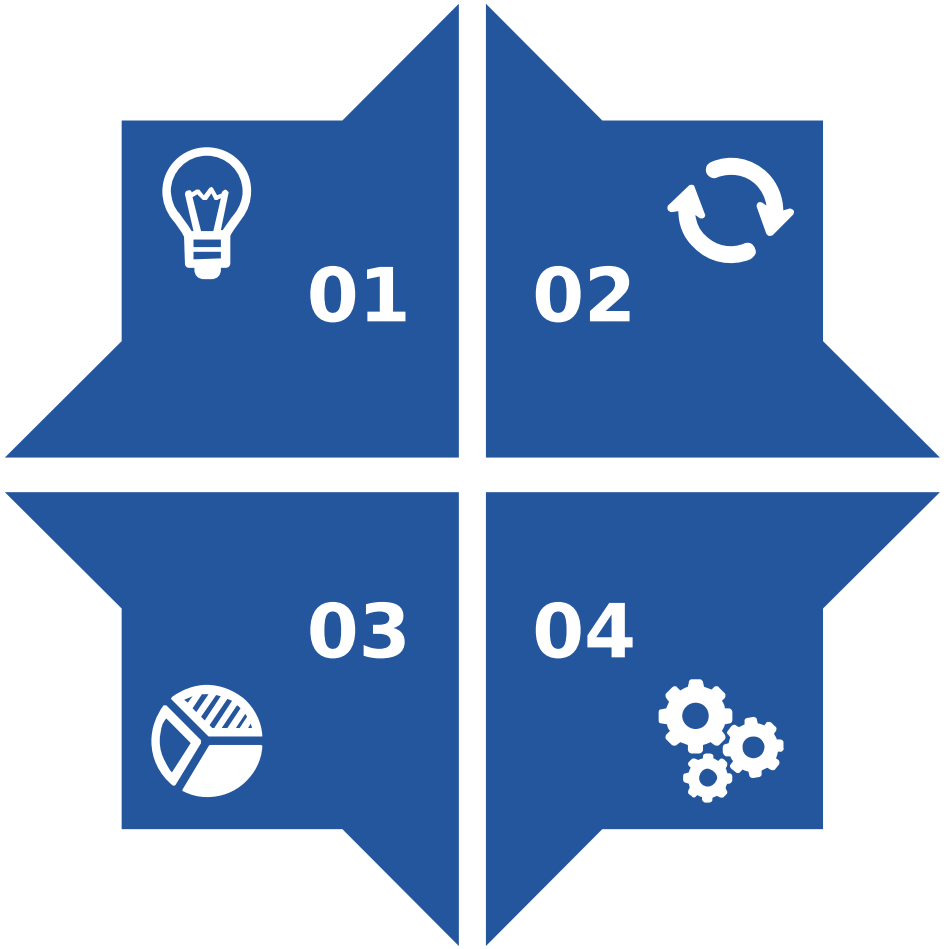
课程《人工智能原理与技术》

同济大学 TONGJI UNIVERSITY

第八周 机器学习与无监督学习

同济大学 TONGJI UNIVERSITY

目录



无监督学习

聚类算法

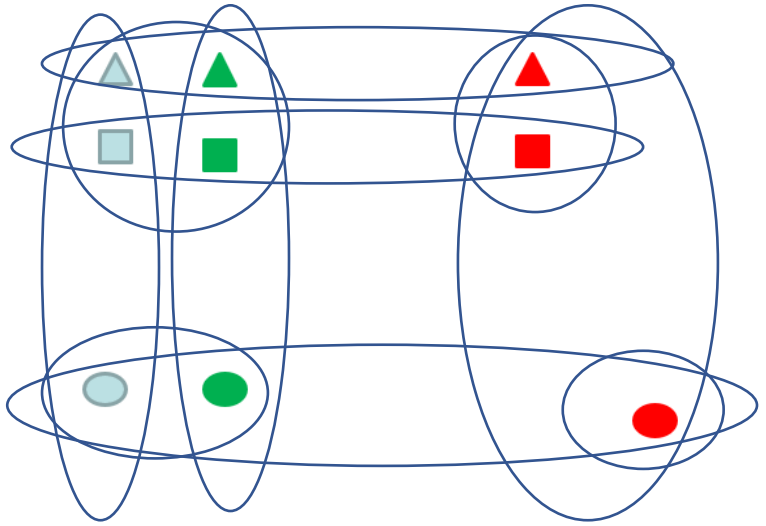
特征降维

演化算法

无监督学习的基本概念

- 从没有“输入 - 输出”关系对的数据中学习模式 / 特征
  - 仅有输入，没有输出
  - 没有标签 / 没有目标
  - 有时也被称为知识发现
  - 希望学习出一些有用的模式

例：划分法下列图形
按照：距离？形状？颜色？



监督学习 VS 无监督学习

监督学习
- 目标明确
- 需要带标签数据
- 效果容易评估

无监督学习
- 目标不明确
- 不需要带标签数据
- 效果较难评估

无监督学习的使用场景

案例 1： 发现异常

有很多违法行为都需要＂洗钱＂，这些洗钱行为跟普通用户的行为是不一样的，到底哪里不一样?

如果通过人为去分析是一件成本很高很复杂的事情，我们可以通过这些行为的特征对用户进行分类，就更容易找到那些行为异常的用户，然后再深入分析他们的行为到底哪里不一样，是否属于违法洗钱的范畴。

通过无监督学习，我们可以快速把行为进行分类，虽然我们不知道这些分类意味着什么，但是通过这种分类，可以快速排出正常的用户，更有针对性的对异常行为进行深入分析。

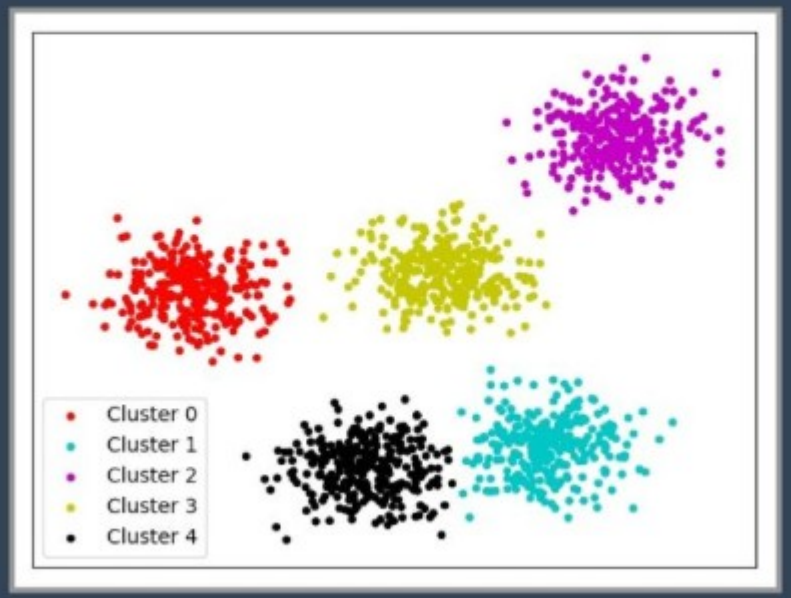
案例 2： 用户细分

这个对于广告平台很有意义，我们不仅把用户按照性别、年龄、地理位置等维度进行用户细分，还可以通过用户行为对用户进行分类。

通过很多维度的用户细分，广告投放可以更有针对性，效果也会更好。

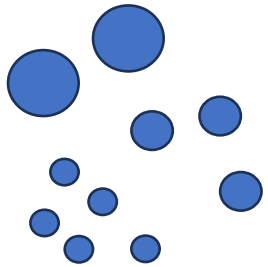
无监督学习的使用场景

借助无监督学习细分用户



借助无监督学习给用户做推荐

常见的无监督学习算法

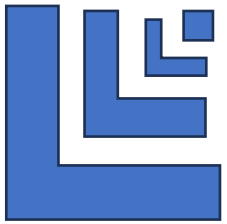


聚类：

一种典型的无监督学习算法，主要用于将相似的样本自动归到一个类别中。

在聚类算法中根据样本之间的相似性，将样本划分到不同的类别中，对于不同的相似度计算方法，会得到不同的聚类结果，常用的相似度计算方法有欧式距离法。

常见算法： K-means ， DBSCN ，层次聚类



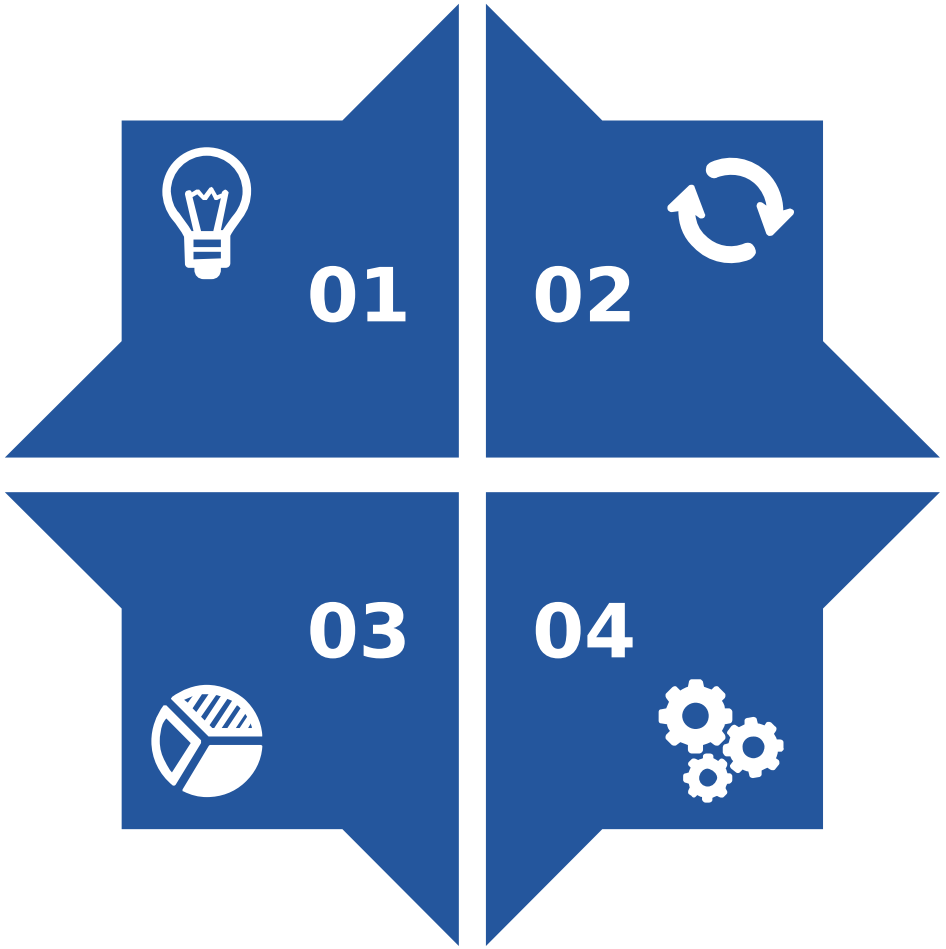
降维：

降维方法旨在将高维数据映射到低维空间，同时尽可能保留数据的重要信息。降维的主要目的是减少数据的复杂性、去除冗余特征、提高计算效率，并有助于可视化和模型性能的提升。

常见算法：线性判别分析、主成分分析

同济大学 TONGJI UNIVERSITY

目录



无监督学习

聚类算法

特征降维

演化算法

“物以类聚，人与群分”

聚类算法的基本概念

- 聚类算法是一种无监督学习方法，旨在将数据集中的对象划分为若干个类别或簇，使得同一类别内的对象相似度较高，不同类别之间的相似度较低。聚类分析在多个领域均有广泛应用，如市场细分、生物学研究、图像分析、社交网络分析等。
- 聚类算法的基本步骤包括选择相似度度量方法、选择聚类算法、确定聚类数目和评估聚类分析结果。

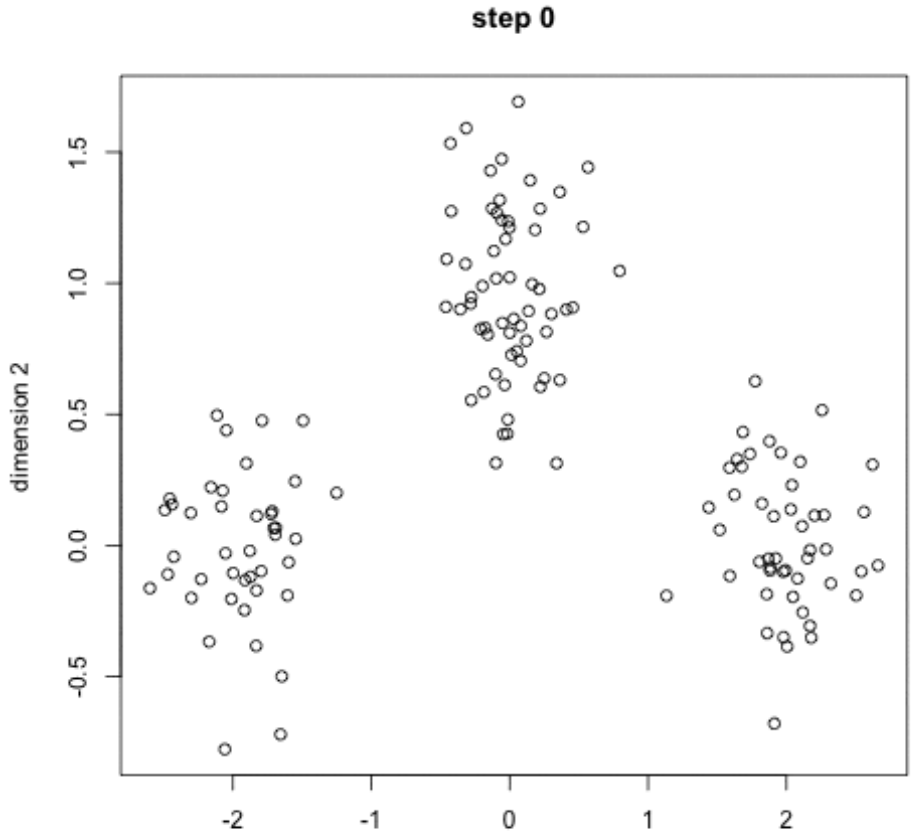
K-means

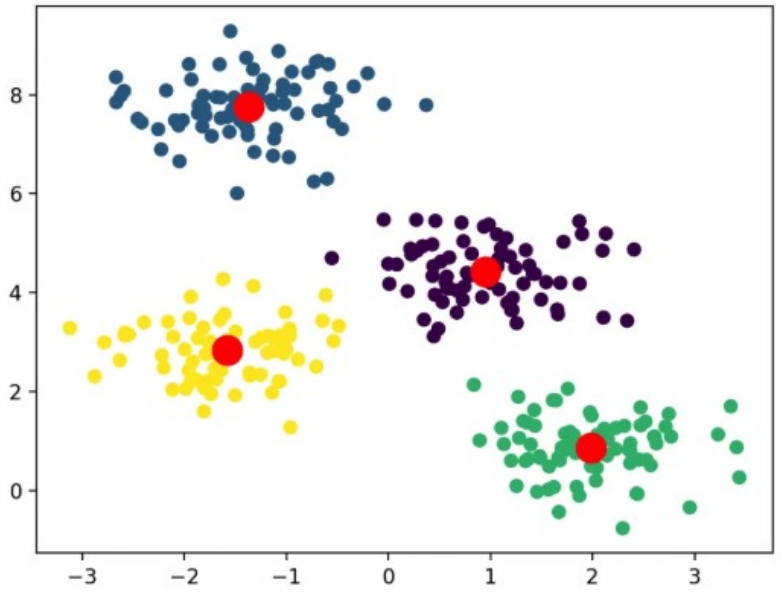
DBSCAN

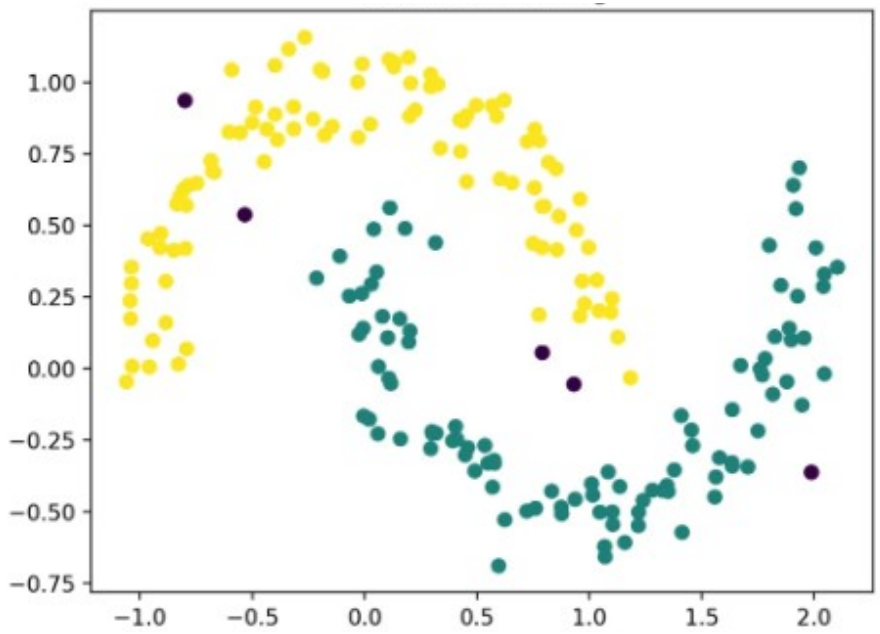
层次聚类

谱聚类

聚类算法的过程视图


step 0
dimension 2
1.5
1.0
0.5
0.0
-0.5
-2
-1
0
1
2


8
6
4
2
0
−3
−2
−1
0
1
2
3


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

K 均值聚类

- k-means 算法的目标是将n个d维数据 $\{x\_i,i=1,\ldots,n\}$ 划分为K个聚簇，使得簇内方差最小化。由于原始数据可能的聚类结果数量巨大，要求一个特定的聚类算法总是能达到最优是不切实际的。所以， k-means 算法找到的是一个“局部”最优，即没有任何其他的聚类结果，能够让簇内的方差更小，但不能保证找到全局最优 [Hartigan 1979] 。 k-means 同时也是一个易受初始值影响的迭代算法，可以用不同的初始值重复几次，以达到上述的“局部”最优，常用的初始化方法包括 Forgy 和 Random Partition[Hamerly 2002] 。

K 均值聚类

1. 初始化聚类质心

初始化 K 个聚类质心 $C=\{c_1, c_2, \cdots, c_K\}$，$c_j \in \mathbb{R}^d$（$1 \leqslant j \leqslant K$）每个聚类质心 c_j 所在的集合记为 G_j。

2. 对数据进行聚类

将每个待聚类数据放入唯一一个聚类集合中。首先通过既定的相似度 / 距离函数计算每个待聚类数据和 K 个聚类质心的距离。以最常用的欧氏距离为例，计算 x_i 和 c_j 之间的欧氏距离：

$$\operatorname{dist}(x_i, c_j)=\sqrt{\sum_{o=1}^{d}(x_{i,o}-c_{j,o})^2} \quad (1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant K)$$

将每个 x_i 放入与之距离最近的聚类质心所在的聚类集合中，即 $\arg\min_{c_j \in C} \operatorname{dist}(x_i, c_j)$。

3. 更新聚类质心

根据聚类结果更新聚类质心，即根据每个聚类集合中所包含的数据，求均值得到该聚类集合新的质心，即

$$c_j=\frac{1}{|G_j|}\sum_{x_i \in G_j} x_i$$

这也是 k–means 名字的来源。均值操作简单高效，这让 k–means 算法可以轻易地被应用在较大规模数据上。不过，均值操作也带来了许多问题，例如易受异常值影响。下面的分析中会对此进行总结。

4. 迭代

算法迭代，即继续根据新的聚类质心，按照欧氏距离大小，将每个待聚类数据放入唯一一个聚类集合中，再根据新的聚类结果更新聚类质心。

聚类迭代终止的判断条件不唯一，经常使用的方法有以下两种：

（1）已经达到迭代次数上限。

（2）前后两次迭代中，聚类质心保持不变。

K 均值聚类的目标

目标函数： $\arg\min_{G} \sum_{i=1}^{k} \sum_{x \in G_i} ||x - G_i||^2 = \arg\min_{G} \sum_{i=1}^{k} |G_i| \, Var \, G_i$

第i个类簇的方差：$\mathrm{var}(G_i) = \frac{1}{|G_i|} \sum_{x \in G_i} ||x - G_i||^2$

- 聚类算法的目标都是得到一个聚类结果，最小化类内距离（或最大化类内相似度），而最大化类间距离（或最小化类间相似度）。
- *k*-means 聚类就是通过最小化聚簇内的数据方差来实现最大化类内相似度的，即最小化每个类簇方差，使得最终聚类结果中每个聚类集合所包含的数据呈现出的差异性最小。

K 均值聚类算法流程

算法步骤

1.初始化：随机选择 K 个数据点作为初始簇中心。

2.分配：将每个数据点分配给最近的簇中心。

3.更新：重新计算每个簇的中心（即簇内所有点的均值）。

4.迭代：重复步骤 2 和 3 直到簇中心不再发生变化或达到预设的迭代次数。

Python 伪代码：

```
def kmeans(data, k, max_iterations=100):
    # 步骤 1: 初始化 - 随机选择 k 个数据点作为初始簇中心
    centers = initialize_centers(data, k)

    for iteration in range(max_iterations):
        # 步骤 2: 分配 - 将每个数据点分配给最近的簇中心
        labels = assign_clusters(data, centers)

        # 步骤 3: 更新 - 重新计算每个簇的中心
        new_centers = update_centers(data, labels, k)

        # 检查收敛条件：如果簇中心不再变化，则停止迭代
        if convergence_check(centers, new_centers):
            break

        # 更新簇中心
        centers = new_centers

    return labels, centers
```

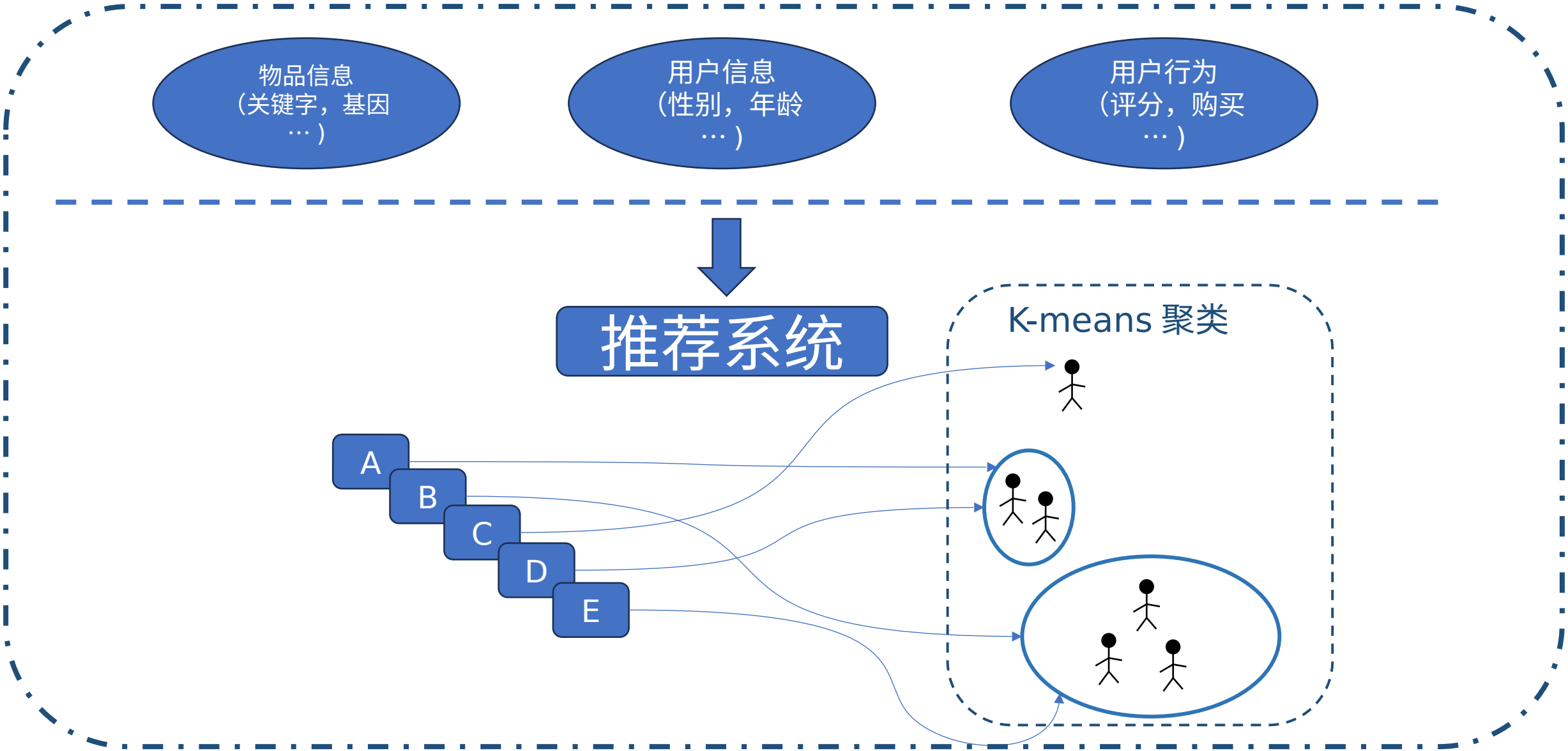
```
def initialize_centers(data, k):// 随机选择 k 个数据点作为初始簇中心
indices = random.sample(range(len(data)), k)
    centers = [data[i] for i in indices]
    return centers

def assign_clusters(data, centers): // 将每个数据点分配给最近的簇中心
labels = []
    for point in data:
        # 计算点到每个簇中心的距离
        distances = [distance(point, center) for center in centers]
        # 找到最近的簇中心
        closest_center_index = np.argmin(distances)
        labels.append(closest_center_index)
    return labels

def update_centers(data, labels, k): // 重新计算每个簇的中心（簇内所有点的均值）
new_centers = []
    for i in range(k):
        # 获取属于当前簇的所有点
        cluster_points = [point for point, label in zip(data, labels) if label == i]
        # 计算簇内点的均值作为新的簇中心
        new_center = np.mean(cluster_points, axis=0)
```

K-means 与推荐引擎

推荐系统相关概念

- 推荐系统是现代互联网公司的核心业务，它通过分析用户行为、内容特征等多种信息，为用户提供个性化的内容推荐。随着数据规模的不断增加，传统的推荐算法已经无法满足业务需求，因此需要开发更高效、准确的推荐算法。
- 在推荐系统中， K-Means 算法可以用于用户行为特征的聚类，从而改善用户个性化推荐的准确性。具体应用场景包括：
  - 用户群集：根据用户的行为特征，将用户划分为多个群集，以便为每个群集内的用户提供更个性化的推荐。
  - 内容群集：根据内容的特征，将内容划分为多个群集，以便为用户推荐与其兴趣相近的内容。
  - 异常行为检测：通过对用户行为特征进行聚类，可以发现异常行为，进而进行异常行为的检测和处理。

推荐示意：



用户聚类

- 类别信息：性别、年龄、职业等等，
- 特征处理：使用 one-hot 把类别信息变成 0 、 1 的值
- 行为列表：播放、购买等等，
- 特征处理：因为是变长的，所以使用 embedding 的技术，转变成一个定长的密集向量。

特征工程之后，把 one-hot 向量列表和 embedding 向量列表进行拼接，成一个大的向量列表（Vector Assembler），里面都是数字，把 Vector Assembler 使用聚类算法。不断迭代计算用户之间特征的相似度，使得各个用户类别之间距离平方和最小。

用户属性

$$\text{目标函数：}\ \arg\min_{G}\sum_{i=1}^{k}\sum_{x\in G_i}||x-G_i||^2=\arg\min_{G}\sum_{i=1}^{k}|G_i|\,Var\,G_i$$

embedding：把有序列表输出成定长向量，每一个向量的值是一个数字，这样不同人的行为列表就可以通过向量直接计算相似度。

同济大学 TONGJI UNIVERSITY

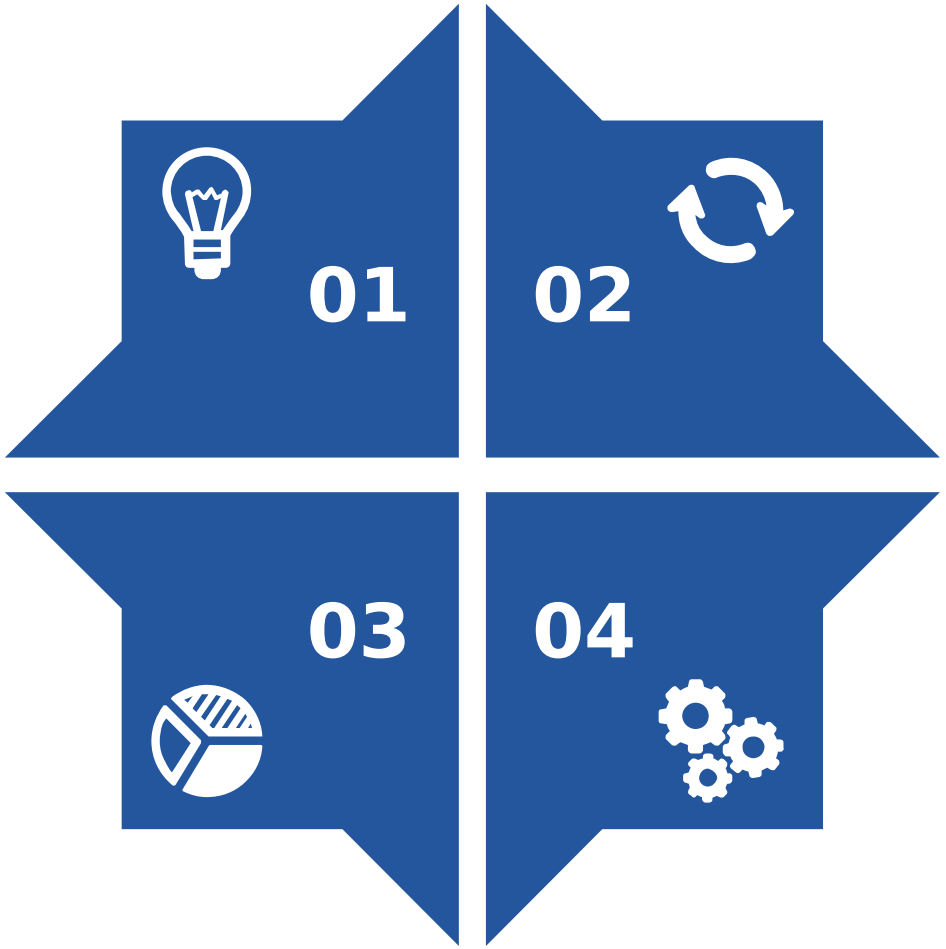
目录

无监督学习

聚类算法

特征降维

演化算法



线性判别分析的基本概念

- 线性判别分析 (linear discriminant analysis， LDA) 是一种基于监督学习的降维方法，也称为 Fisher 线性判别分析（fisher's discriminant analysis， FDA) [Fisher 1936] 。对于一组具有标签信息的高维数据样本， LDA 利用其类别信息，将其线性投影到一个低维空间上，在低维空间中同一类别样本尽可能靠近，不同类别样本尽可能彼此远离。

- “类内方差小、类间间隔大”
- 《论语 · 子路》：君子和而不同，小人同而不和

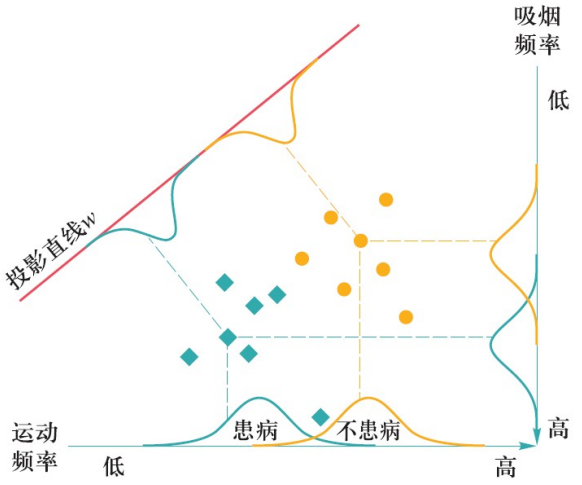


图 4.4 两个类别数据所对应的不同投影方式

正方形表示患病个体、圆形表示不患病个体

线性判别分析（LDA）

假设样本集为$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$，样本$\boldsymbol{x}_i \in \mathbb{R}^d$的类别标签为$y_i$。其中，$y_i$的取值范围是$\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$，即一共有$K$类样本。

定义$\boldsymbol{X}$为所有样本构成集合、N_i为第i个类别所包含样本个数、X_i为第i类样本的集合、$\boldsymbol{m}$为所有样本的均值向量、$\boldsymbol{m}_i$为第i类样本的均值向量。Σ_i为第i类样本的协方差矩阵，其定义为:

$$\Sigma_i = \sum_{\boldsymbol{x} \in \boldsymbol{X_i}} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

先来看$K = 2$的情况，即二分类问题。在二分类问题中，训练样本归属于$\mathcal{C}_1$或$\mathcal{C}_2$两个类别，并通过如下的线性函数投影到一维空间上:

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} \ (\boldsymbol{w} \in \mathbb{R}^n)$$

投影之后类别$\mathcal{C}_1$的协方差矩阵s_1为:

$$s_1 = \sum_{\boldsymbol{x} \in \mathcal{C}_1} (\boldsymbol{w}^T \boldsymbol{x} - \boldsymbol{w}^T \boldsymbol{m}_1)^2 = \boldsymbol{w}^T \sum_{\boldsymbol{x} \in \mathcal{C}_1} [(\boldsymbol{x} - \boldsymbol{m}_1)\,(\boldsymbol{x} - \boldsymbol{m}_1)^T] \boldsymbol{w}$$

同理可得到投影之后类别$\mathcal{C}_2$的协方差矩阵s_2。

线性判别分析（LDA）

可以看到，投影后两个协方差矩阵s_1和s_2分别为$\boldsymbol{w}^T\Sigma_1\boldsymbol{w}$和$\boldsymbol{w}^T\Sigma_2\boldsymbol{w}$。由于将数据样本投影到了一维空间直线上，$s_1$和$s_2$都是实数。$s_1$和$s_2$可用来衡量同一类别数据样本之间“分散程度”。为了使得归属于同一类别的样本数据在投影后的空间中尽可能靠近，需要最小化s_1+s_2取值。

在投影之后的空间中，归属于两个类别的数据样本中心可分别如下计算：

$$m_1 = \boldsymbol{w}^T\boldsymbol{m}_1, \quad m_2 = \boldsymbol{w}^T\boldsymbol{m}_2$$

这样，就可以通过$\|m_2 - m_1\|_2^2$来衡量不同类别之间的距离。为了使得归属于不同类别的样本数据在投影后空间中尽可能彼此远离，需要最大化过$\|m_2 - m_1\|_2^2$取值。

线性判别分析（LDA）：优化目标

最大化的目标$J(\boldsymbol{w})$，定义如下：

$$J(\boldsymbol{w}) = \frac{\|m_2 - m_1\|_2^2}{s_1 + s_2}$$

可以把上述式子右边改写成与$\boldsymbol{w}$相关的式子：

$$J(\boldsymbol{w}) = \frac{\left\|\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)\right\|_2^2}{\boldsymbol{w}^T\Sigma_1\boldsymbol{w} + \boldsymbol{w}^T\Sigma_2\boldsymbol{w}} = \frac{\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w}}{\boldsymbol{w}^T(\Sigma_1 + \Sigma_2)\boldsymbol{w}} = \frac{\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}}$$

其中，$\boldsymbol{S}_b$称为类间散度矩阵(between-class scatter matrix)，即衡量两个类别均值点之间的“分离”程度，可定义如下：

$$\boldsymbol{S}_b = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$$

$\boldsymbol{S}_w$则称为类内散度矩阵(within-class scatter matrix)，即衡量每个类别中数据点的“分离”程度，可定义如下：$\boldsymbol{S}_w = \Sigma_1 + \Sigma_2$

$$\boldsymbol{S}_w = \Sigma_1 + \Sigma_2$$

线性判别分析（LDA）：优化目标

$$J(\boldsymbol{w}) = \frac{\left\|\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)\right\|_2^2}{\boldsymbol{w}^T\Sigma_1\boldsymbol{w} + \boldsymbol{w}^T\Sigma_2\boldsymbol{w}} = \frac{\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w}}{\boldsymbol{w}^T(\Sigma_1 + \Sigma_2)\boldsymbol{w}} = \frac{\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}}$$

由于$J(\boldsymbol{w})$的分子和分母都是关于$\boldsymbol{w}$的二项式，因此最后的解只与$\boldsymbol{w}$的方向有关，与$\boldsymbol{w}$的长度无关，因此可令分母$\boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w} = 1$，然后用拉格朗日乘子法来求解这个问题。拉格朗日乘子法就是求在某个/某些约束条件下的函数极值的方法，其主要思想是将约束条件函数与原函数联立，从而求出使原函数取得极值时各个变量的解。

线性判别分析（LDA）：优化目标

$$J(\boldsymbol{w}) = \frac{\left\|\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)\right\|_2^2}{\boldsymbol{w}^T\Sigma_1\boldsymbol{w} + \boldsymbol{w}^T\Sigma_2\boldsymbol{w}} = \frac{\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w}}{\boldsymbol{w}^T(\Sigma_1 + \Sigma_2)\boldsymbol{w}} = \frac{\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}}$$

上述带约束条件（即$\boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w} - 1 = 0$）的函数极大值（即$\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}$取值最大）优化问题所对应拉格朗日函数为：

$$L(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w} - \lambda(\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} - 1)$$

对$\boldsymbol{w}$求偏导并使其求导结果为零，可得$\boldsymbol{S}_w{}^{-1}\boldsymbol{S}_b\boldsymbol{w} = \lambda\boldsymbol{w}$。由此可见，$\lambda$和$\boldsymbol{w}$分别是$\boldsymbol{S}_w{}^{-1}\boldsymbol{S}_b$的特征根和特征向量，$\boldsymbol{S}_w{}^{-1}\boldsymbol{S}_b\boldsymbol{w} = \lambda\boldsymbol{w}$也被称为Fisher线性判别（Fisher linear discrimination）。

因为$\boldsymbol{S}_b = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$，令实数$\lambda_w = (\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w}$，那么$\boldsymbol{S}_b\boldsymbol{w} = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w} = (\boldsymbol{m}_2 - \boldsymbol{m}_1) \times \lambda_w$，将其代入Fisher线性判别，可得：

$$\boldsymbol{S}_w{}^{-1}\boldsymbol{S}_b\boldsymbol{w} = \boldsymbol{S}_w{}^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1) \times \lambda_w = \lambda\boldsymbol{w}$$

由于对$\boldsymbol{w}$的放大和缩小操作不影响结果，因此可约去上式中的未知数λ和λ_w，得到：$\boldsymbol{w} = \boldsymbol{S}_w{}^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$

注：上述推导中利用了如下公式：如果$\mathbf{A}$是实对称矩阵，则$\boldsymbol{x}^T\mathbf{A}\boldsymbol{x}$对$\boldsymbol{x}$的导数为$\frac{d\boldsymbol{x}^T\mathbf{A}\boldsymbol{x}}{\boldsymbol{x}} = 2\mathbf{A}\boldsymbol{x}$

线性判别分析（LDA）算法流程

算法流程：

- 输入数据 X 和标签 y
- 计算类内散度矩阵 Sw
- 计算类间散度矩阵 Sb
- 求解 Sw^{-1} Sb 的特征值和特征向量，获得投影矩阵
- 将数据 X 投影到低维空间： X_lda = X @ W

Python 伪代码：

```
def linear_discriminant_analysis(X, y,
n_components):

# 步骤 1：计算类内散度矩阵（Sw）和类间散度矩阵（Sb）
    Sw = compute_within_class_scatter(X, y)
    Sb = compute_between_class_scatter(X, y)

    # 步骤 2：计算 Sw^{-1} * Sb 的特征值和特征向量
    eigenvalues, eigenvectors =
np.linalg.eig(np.linalg.inv(Sw) @ Sb)

    # 步骤 3：按特征值从大到小排序，选择前
n_components 个特征向量
    sorted_indices = np.argsort(eigenvalues)[::-1]
    W = eigenvectors[:,
sorted_indices[:n_components]]

    # 步骤 4：将数据投影到低维空间
    X_lda = X @ W

    return W, X_lda
```

```
def compute_within_class_scatter(X, y):
    """
    计算类内散度矩阵 Sw
    Sw = Σ (Σ (x - μ_i)(x - μ_i)^T) for each class i
    """
    n_features = X.shape[1]
    Sw = np.zeros((n_features, n_features))
    classes = np.unique(y)

    for c in classes:
        X_c = X[y == c]
        mean_c = np.mean(X_c, axis=0)
        Sw += (X_c - mean_c).T @ (X_c - mean_c)

    return Sw

def compute_between_class_scatter(X, y):
    """
    计算类间散度矩阵 Sb
    Sb = Σ n_i * (μ_i - μ)(μ_i - μ)^T for each class i
    """
    n_features = X.shape[1]
    Sb = np.zeros((n_features, n_features))
    mean_total = np.mean(X, axis=0)
    classes = np.unique(y)

    for c in classes:
        X_c = X[y == c]
        n_c = X_c.shape[0]
        mean_c = np.mean(X_c, axis=0)
        Sb += n_c * (mean_c - mean_total).reshape(-1, 1) @ (mean_c -
mean_total).reshape(1, -1)

    return Sb
```

同济大学 TONGJI UNIVERSITY

线性判别分析（LDA）的应用场景

同济大学 TONGJI UNIVERSITY

1. 医学诊断

➢ 疾病诊断

在医学领域，患者的各项生理指标（如血压、血糖、心率等）构成了一个高维的特征空间。 LDA 可以用于分析这些指标，找出最能区分健康人和患者的特征组合。例如，在诊断心脏病时，通过对大量患者和健康人的生理数据进行 LDA 分析，确定哪些指标的组合对于判断是否患有心脏病最为关键。这样医生可以根据这些关键指标更准确地进行诊断。

➢ 癌症分类

癌症有多种类型，不同类型的癌症在基因表达、细胞形态等方面存在差异。 LDA 可以对这些生物学特征进行分析，将不同类型的癌症样本进行有效的分类。例如，通过分析肿瘤组织的基因表达数据， LDA 可以找到最能区分不同癌症亚型的基因特征，为癌症的精准治疗提供依据。

2. 金融分析

➢ 信用风险评估

银行在评估客户的信用风险时，会考虑多个因素，如客户的收入、负债情况、信用历史等 LDA 可以将这些高维的客户信息投影到一个低维空间，根据投影后的结果将客户分为不同的信用等级。例如，将客户分为低风险、中风险和高风险三类，银行可以根据这些分类来决定是否给予贷款以及贷款的额度和利率。

➢ 股票市场预测

股票市场受到多种因素的影响，如公司财务状况、宏观经济指标、行业发展趋势等。 LDA 可以用于分析这些因素，将股票分为不同的类别，如上涨潜力大、稳定和下跌可能性高的股票。投资者可以根据这些分类来制定投资策略。

主成分分析（principal component analysis）

- 主成分分析（principal component analysis）是一种特征降维方法，在消除数据噪声、冗余等方面具有广泛应用。主成分分析也被称为 KL 变换（Karhunen–Loève transform, KLT）、霍林特变换（Hotelling transform）或者本征正交分解（proper orthogonal decomposition, POD）等。
- 顾名思义，主成分分析即通过分析找到数据特征的主要成分，使用这些主要成分来代替原始数据。这样一方面可以加深对数据本身的理解（认识到数据的主要成分）；另一方面，简化后的数据（主要成分）在用于下游的其他任务时，有着噪声少、易于处理计算的特点。 这种使用主要成分而不是全部数据的思想也与“如无必要，勿增实体”的奥卡姆剃刀原则相符，体现了“简单有效原理”。
- 主成分分析要求“降维后的结果要保持原始数据的原有结构”，例如，对于图像数据，要求保持视觉对象区域构成的空间分布；对于文本数据，要求保持单词之间的（共现）相似或不相似的特性。更准确地说，主成分分析要求最大限度保持原始高维数据的总体方差结构。

主成分分析（principal component analysis）

方差 方差描述了样本数据的波动程度，数值上等于各个数据与样本均值之差的平方和之平均数，假设有n个数据，记为$X = \{x_i\}(i = 1, \dots, n)$，那么样本方差（sample variance）即为

$$\mathrm{var}(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - u)^2$$

其中u是样本均值，$u = \frac{1}{n}\sum_{i=1}^{n} x_i$。上述样本方差公式里分母为$\mathrm{n}-1$的目的是为了让对方差的估计是无偏估计 (unbiased estimator)。

协方差 协方差衡量了两个变量之间的相关度，假设有两个变量，观察到不同时刻两个变量的取值，记为$(X, Y) = \{(x_i, y_i\}\ (i = 1, \dots, n)$，那么两个变量的协方差为：

$$\mathrm{cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y))$$

其中$E(X)$和$E(Y)$分别是X和Y的样本均值，分别定义如下$E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i, E(Y) = \frac{1}{n}\sum_{i=1}^{n} y_i$。

主成分分析（principal component analysis）

皮尔逊相关系数 由于协方差会受到变量取值尺度的影响，可以通过皮尔逊相关系数（Pearson Correlation coefficient）将两组变量之间的关联度规整到一定的取值范围内。皮尔逊相关系数定义如下：

$$\mathrm{corr}(X,Y)=\frac{\mathrm{Cov}(X,Y)}{\sqrt{\left(\mathrm{Var}(X)\mathrm{Var}(Y)\right)}}=\frac{\mathrm{Cov}(X,Y)}{\sigma_x\sigma_y}$$

其中$\sigma_x\sigma_y$分别为X和Y的标准差。皮尔逊相关系数所具有的性质如下：

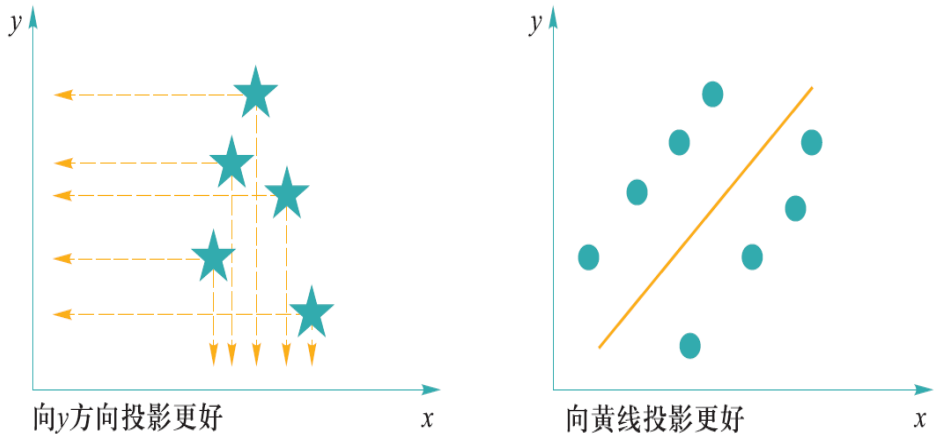
$$|\mathrm{corr}(X,Y)|\leq 1$$

主成分分析（principal component analysis）

主成分分析（Principle Component Analysis）的思想是将d维特征数据映射到l维空间（一般$d \gg l$），去除原始数据之间的冗余性（通过去除相关性手段达到这一目的）。将原始数据向这些数据方差最大的方向进行投影。一旦发现了方差最大的投影方向，则继续寻找保持方差第二的方向且进行投影。其目标是使得数据每一维的方差都尽可能大。

例：

如左图所示，向y方向投影（使得二维数据映射为一维）就比向x方向投影结果在降维这个意义上而言要好；右图则是黄线方向投影要好。这样的投影结果更好的保留了未降维前数据的离散程度。



算法 4.2 主成分分析

输入： n 个 d 维样本数据所构成的矩阵 X，降维后的维数 l

输出： 映射矩阵 $W=\{w_1, w_2, \cdots, w_l\}$

算法步骤：

（1）对于每个样本数据 x_i 进行中心化处理：$x_i = x_i - \mu$，$\mu = \frac{1}{n}\sum_{i=1}^{n} x_j$。

（2）计算原始样本数据的协方差矩阵：$\Sigma = \frac{1}{n-1} X^T X$

（3）对协方差矩阵 Σ 进行特征值分解，对所得特征根进行排序：$\lambda_1 \geqslant \lambda_2 \geqslant \lambda_l$。

（4）取前 l 个最大特征根所对应的特征向量 $w_1, w_2, \cdots, w_l$ 组成映射矩阵。

同济大学 TONGJI UNIVERSITY

Python 伪代码

```
def pca(data, n_components):
    # 步骤 1：数据标准化
    # 计算每列的均值
    mean = np.mean(data, axis=0)
    standardized_data = data – mean

    # 步骤 2：计算协方差矩阵
    covariance_matrix = np.cov(standardized_data,
rowvar=False)

    # 步骤 3：计算协方差矩阵的特征值和特征向量
    eigenvalues, eigenvectors =
```

```
# 步骤 4：对特征值进行排序，并选择前 n_components 个特征向量
    # 获取特征值的索引并按降序排序
    sorted_indices = np.argsort(eigenvalues)[::-1]
    top_eigenvectors = eigenvectors[:,
sorted_indices[:n_components]]

    # 步骤 5：数据投影
    # 将标准化后的数据投影到选定的特征向量上
    reduced_data = np.dot(standardized_data, top_eigenvectors)
    return reduced_data
```

主成分分析与人脸识别

- 特征人脸是基于外观的人脸识别方法，其目的是捕捉人脸图像集合中的特征信息，并使用该信息对各个人脸图像进行编码和比较。换句话说，特征人脸提取了人脸图像的面部信息，这些信息可能与人的直觉相符，如眼睛、鼻子和嘴唇的大小、形状特征。这样，特征人脸就可以有效地表现原始面部图像，减少计算和空间复杂度。
- 使用主成分分析的手段表示特征人脸的思想最先由 Sirovich 和 Kirby 提出，其本质是使用一组特征向量的线性组合来表示原始人脸，进而实现人脸识别。
- 特征人脸方法用一种称为“特征人脸 (eigenface)” 的特征向量按照线性组合形式来表达每一张原始人脸图像，进而实现人脸识别。

主成分分析与人脸识别

假设总共有n张灰度人脸图像，每张人脸图像Γ_i的分辨率为$d \times d$。如果$d = 32$，即人脸图像的分辨率为32×32，人脸图像在计算机中可以表示为一个32×32的像素矩阵，如果将该矩阵铺平，可以转换为一个1024维的列向量，如图4.6所示

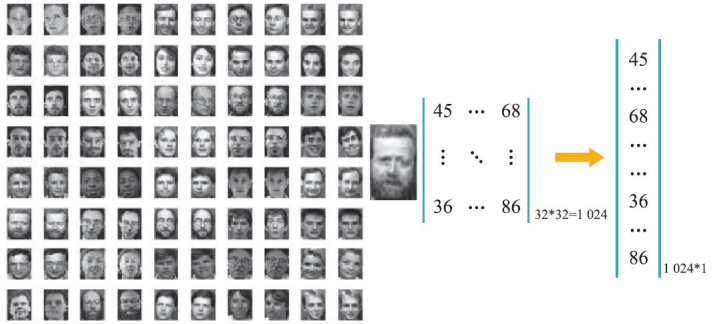


图 4.6　二维灰度图像的向量化表示

该 1024 维的列向量是原始灰度图像的表达。但是直觉表明，人脸具有一定的拓扑结构，像素点之间有较强的空间位置关系，也就是说，可以使用一个低维向量来表达原始图像大部分的信息。这样就可以使用前面讲述的主成分分析算法来实现降维。不过在特征维度较高的情况下，主成分分析算法暴力求解特征向量是一个耗时操作。这里介绍一种新的矩阵分解方法——奇异值分解（ singular value decomposition ， SVD ）来实现主成分分析，对原始数据进行降维。

主成分分析与人脸识别

奇异值分解将矩阵 A 分解为三个子矩阵，即 $A=UDV^{\mathrm{T}}$，其中矩阵 U 和矩阵 V 都是酉矩阵，即满足 $UU^{\mathrm{T}}=VV^{\mathrm{T}}=I$。矩阵 D 是对角矩阵，它的每一项都是正实数。与特征分解不同，奇异值分解不要求被分解的矩阵 A 是方阵，假如矩阵 A 的维度是 $n\times d$，则矩阵 U 的维度为 $n\times n$，矩阵 D 的维度为 $n\times d$，矩阵 V^{T} 的维度为 $d\times d$，如图 4.7 所示。

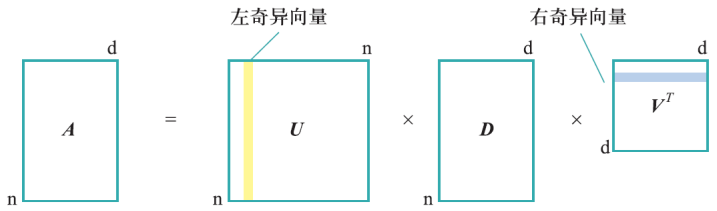


图 4.7　奇异值分解示意

那么如何求解分解后的三个子矩阵呢？容易发现：

$$AA^{\mathrm{T}}=UDV^{\mathrm{T}}(UDV^{\mathrm{T}})=UDV^{\mathrm{T}}VD^{\mathrm{T}}U^{\mathrm{T}}=UDD^{\mathrm{T}}U^{\mathrm{T}}=UD^2U^{\mathrm{T}}$$

$$\Rightarrow (AA^{\mathrm{T}})U=UD^2$$

因此，矩阵 U 即为矩阵 AA^{T} 所有特征向量构成的矩阵；同理，矩阵 V 即为矩阵 $A^{\mathrm{T}}A$ 所有特征向量构成的矩阵。同时，矩阵 D 就等于矩阵 AA^{T} 或矩阵 $A^{\mathrm{T}}A$ 的所有特征值开方后组成的对角矩阵；或者也可以使用矩阵 V 和矩阵 U，通过式子 $U^{\mathrm{T}}AV=D$ 得到矩阵 D。

主成分分析与人脸识别

首先计算所有 n 张灰度人脸图像向量的均值人脸向量，即 $\boldsymbol{\Psi}=\frac{1}{n}\sum_{i=1}\boldsymbol{\Gamma}_i$。得到均值人脸向量后，用每张人脸图像向量减去均值人脸向量，即进行中心化处理，得到处理后的人脸向量，记为 $\boldsymbol{\phi}_i=\boldsymbol{\Gamma}_i-\boldsymbol{\Psi}(1\leqslant i\leqslant n)$。

现在，所有处理后的人脸向量可以表示为一个 $n\times d^2$ 的矩阵，记为 $\boldsymbol{\Phi}$，$\boldsymbol{\Phi}$ 中的每一行即为中心化处理后的人脸向量。

如果使用主成分分析，需要计算维度为 $d^2\times d^2$ 的特征的协方差矩阵 $\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$，然后计算协方差矩阵的特征向量（右奇异向量），但是当 $d^2>>n$ 时，这就是一个相对耗时甚至不切实际的方法。注意到：

$$U^{\mathrm{T}}A=U^{\mathrm{T}}UDV^{\mathrm{T}}=DV^{\mathrm{T}}\Rightarrow u_i^{\mathrm{T}}A=d_iv_i^{\mathrm{T}}$$

因此，可以使用奇异值分解求取左奇异向量和特征值，然后得到右奇异向量，即求取矩阵 $\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$ 的特征向量，有

$$(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}})u_i=\lambda_iu_i$$

上式两端分别左乘 $\boldsymbol{\Phi}^{\mathrm{T}}$，可得

$$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}u_i)=\lambda_i(\boldsymbol{\Phi}^{\mathrm{T}}u_i)$$

这里的 $\boldsymbol{\Phi}^{\mathrm{T}}u_i$ 就是一个特征人脸，表示为一个维度为 $d^2\times1$ 的列向量。n 个 $d^2\times1$ 的列向量组成矩阵 $U_{d^2\times n}$，该矩阵称为特征人脸空间。需要注意的是，$U_{d^2\times n}$ 中的每一列就是一个特征人脸。

对于先前得到的每幅人脸图像 $\boldsymbol{\phi}_i=\boldsymbol{\Gamma}_i-\boldsymbol{\Psi}(1\leqslant i\leqslant n)$，可以如下得到 $\boldsymbol{\phi}_i$ 在特征人脸空间中的表示：

$$\Omega_i=(U_{d^2\times n})^{\mathrm{T}}(\boldsymbol{\phi}_i)_{d^2\times1}$$

这里将 $\boldsymbol{\phi}_i$ 从 $d\times d$ 大小转换成了 n 维大小的向量，n 维向量中每个值反映了 $\boldsymbol{\phi}_i$ 与每个特征人脸相关性的大小，于是可以用这 n 个相关性系数来表征原始人脸。

这样就实现了将人脸从像素点空间转换到人脸空间，即用特征人脸来表达人脸。要比较两幅人脸图像 $\boldsymbol{\phi}_i$ 和 $\boldsymbol{\phi}_j$ 是否相似，只需比较其对应在特征人脸空间的表示 Ω_i 和 Ω_j 是否相似。

图 4.8　特征人脸向量的可视化

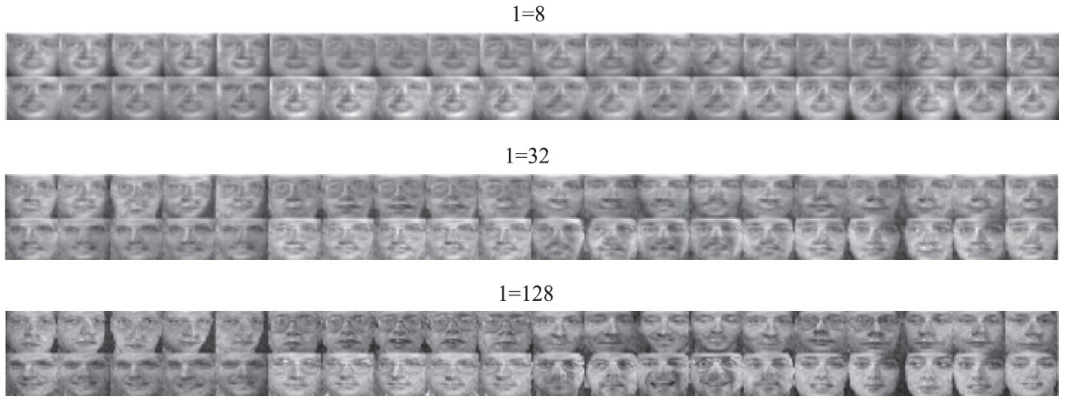
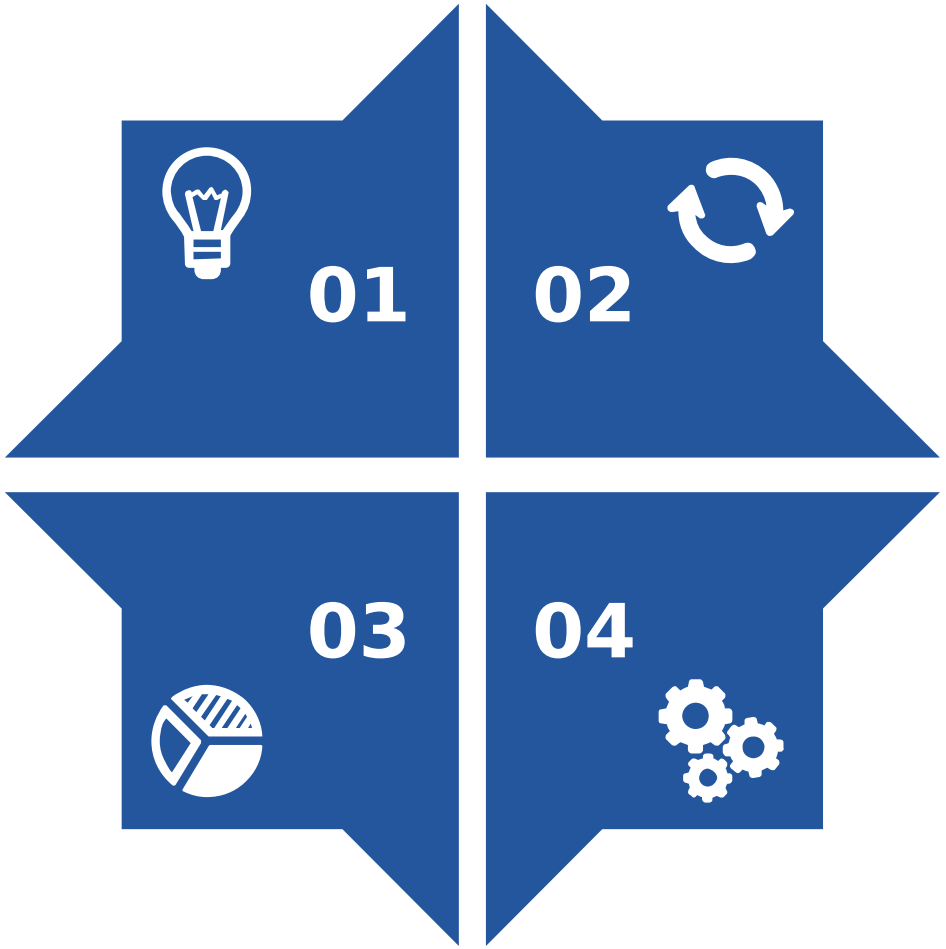


图 4.9　特征人脸重建原始灰度人脸图像

同济大学 TONGJI UNIVERSITY

目录



无监督学习

聚类算法

特征降维

演化算法

演化算法（evolutionary algorithm）

- 演化算法（Evolutionary Algorithms）是指一大类受自然演化启发的启发式随机优化算法，通过“突变重组”和“自然选择”这两种机制来模拟自然演化过程。演化算法有很多种实现方法，如遗传算法(genetic algorithm)、遗传规划(genetic programming，GP)、演化策略(evolutionary strategy)等。
- 遗传算法（Genetic Algorithm, GA）是演化学习的代表性算法之一，其受到达尔文“物竞天择、适者生存”进化论思想启示，引入选择(selection)、交叉(crossover)和变异(mutation)等操作以搜索方式来进行优化求解。遗传算法由密歇根大学的约翰·霍兰德（John H. Holland）和他的同事于二十世纪六十年代在对细胞自动机（cellular automata）进行研究时率先提出。
- 在遗传算法中，对于一个待求解的最优化问题，先给定一定数量候选解（称为个体或染色体），然后让这些染色体所构成种群向最优解进化。在每一次进化过程中，计算每个染色体（即候选解）的适应度（fitness），根据适应度选择合适染色体，随后施以选择和变异来产生新的生命种群，该种群在算法下一次迭代中被作为当前种群。

演化算法（evolutionary algorithm）

- 步骤 1：初始化具有若干规模数目的群体。当前进化代数 Generation=0
- 步骤 2：采用评估函数计算群体中每个染色体的适应值，保存适应值最大的染色体 Best 。
- 步骤 3：采用轮盘赌选择算法对群体中染色体进行选择，产生同样规模的种群。
- 步骤 4：按照概率从种群中选择两对染色体进行交叉操作，即对所选中两对染色体中相应基因片段信息进行交换。交叉所得染色体进入新种群，未交叉操作染色体被复制进入新种群。
- 步骤 5：按照概率对新种群中被选中染色体的若干基因片段进行变异操作。变异后染色体取代原有染色体进入新种群，未发生变异染色体直接进入新群体。
- 步骤 6：计算种群中各个染色体的适应值。倘若种群中染色体最大适应值大于已知 Best 适应值，则取代原有 Best 。
- 步骤 7：当前进化代数 Generation 加 1 。如果 Generation 超过规定的最大进化代数或 Best 达到规定的误差要求，算法结束；否则返回步骤 3 。

Python 伪代码

同济大学
TONGJI UNIVERSITY

```
Generation = 0
# 步骤 1： 初始化具有若干规模数目的群体。当前进化代数 Generation=0
population = initialize_population(population_size, chromosome_length)
Best = None  # 保存适应值最大的染色体

# 步骤 2： 采用评估函数计算群体中每个染色体的适应值，保存适应值最大的染色体 Best。
fitness_scores = evaluate_population(population)
Best = population[np.argmax(fitness_scores)] # 保存适应值最大的染色体

while Generation < max_generations and not convergence_criteria(Best):

    # 步骤 3： 采用轮盘赌选择算法对群体中染色体进行选择，产生同样规模的种群。
    def roulette_wheel_selection(population, fitness_scores):
        total_fitness = sum(fitness_scores)
        probabilities = [fitness / total_fitness for fitness in fitness_scores]
        selected_indices = np.random.choice(len(population), size=len(population),
p=probabilities)
        selected_population = [population[i] for i in selected_indices]
        return selected_population

    selected_population = roulette_wheel_selection(population, fitness_scores)


     # 步骤 4： 按照概率从种群中选择两对染色体进行交叉操作，定义交叉操作函数
    def crossover(parent1, parent2, crossover_rate):
        if random.random() < crossover_rate:
            crossover_point = random.randint(1, len(parent1) - 1)
            child1 = parent1[:crossover_point] + parent2[crossover_point:]
            child2 = parent2[:crossover_point] + parent1[crossover_point:]
            return child1, child2
        else:
            return parent1, parent2

    new_population = []
    for i in range(0, len(selected_population), 2):
        parent1 = selected_population[i]
        parent2 = selected_population[i+1]
        child1, child2 = crossover(parent1, parent2, crossover_rate)
        new_population.extend([child1, child2])

    # 步骤 5： 按照概率对新种群中被选中染色体的若干基因片段进行变异操作
    def mutate(chromosome, mutation_rate):
        mutated_chromosome = []
        for gene in chromosome:
            if random.random() < mutation_rate:
                mutated_chromosome.append(flip_gene(gene)) # 基因变异
            else:
                mutated_chromosome.append(gene)
        return mutated_chromosome

    new_population = [mutate(chromosome, mutation_rate) for chromosome in new_population]

    # 步骤 6： 计算种群中各个染色体的适应值。倘若种群中染色体最大适应值大于已知 Best 适应值，则取代原
有 Best。
    fitness_scores = evaluate_population(new_population)
    current_best_index = np.argmax(fitness_scores)
    if fitness_scores[current_best_index] > evaluate_fitness(Best):
        Best = new_population[current_best_index]

    # 步骤 7： 当前进化代数 Generation 加 1。如果 Generation 超过规定的最大进化代数或 Best 达到规定的
误差要求，算法结束；否则返回步骤 3。
    Generation += 1
```

演化算法举例

已知函数$y = f(x_1, x_2, x_3, x_4) = \frac{1}{x_1^2+x_2^2+x_3^2+x_4^2+8}$,其中$-5 \le x_1, x_2, x_3, x_4 \le 5$，用遗传算法求解函数$y$的最大值及对应变量$x_1$、$x_2$、$x_3$和$x_4$的取值。

步骤1：初始化

假设种群规模数目为5，使用浮点数编码方式构造染色体（即候选解）。每个染色体以四维向量形式表示，对应函数$f(x_1, x_2, x_3, x_4)$中四个变量取值。初始化后得到如下五个染色体：

$$C_1 = (-2.1351, 2.0917, -0.1327, -4.1006)$$

$$C_2 = (1.0152, -3.9811, -2.6638, 3.7535)$$

$$C_3 = (4.0589, 2.1904, -0.1503, 0.0023)$$

$$C_4 = (-3.4098, -3.0714, -0.9008, -4.3712)$$

$$C_5 = (0.2073, 2.9932, -4.0802, 1.8794)$$

演化算法举例

步骤2：计算适应值

如下计算每个染色体适应值 $Eval(C) = f(x_1, x_2, x_3, x_4) = \frac{1}{x_1^2+x_2^2+x_3^2+x_4^2+8}$：

$$Eval(C_1) = f(-2.1351, 2.0917, -0.1327, -4.1006) = 0.0296152$$

$$Eval(C_2) = f(1.0152, -3.9811, -2.6638, 3.7535) = 0.0217087$$

$$Eval(C_3) = f(4.0589, 2.1904, -0.1503, 0.0023) = 0.0341354$$

$$Eval(C_4) = f(-3.4098, -3.0714, -0.9008, -4.3712) = 0.0204169$$

$$Eval(C_5) = f(0.2073, 2.9932, -4.0802, 1.8794) = 0.0268944$$

因此 $Best = C_3, Eval(Best) = 0.0341354$

演化算法举例

步骤3：选择

采用轮盘赌选择算法对染色体进行选择，得到新种群为：

$$C_1' = (1.0152, -3.9811, -2.6638, 3.7535)$$

$$C_2' = (4.0589, 2.1904, -0.1503, 0.0023)$$

$$C_3' = (1.0152, -3.9811, -2.6638, 3.7535)$$

$$C_4' = (1.0152, -3.9811, -2.6638, 3.7535)$$

$$C_5' = (0.2073, 2.9932, -4.0802, 1.8794)$$

演化算法举例

步骤4：交叉

具体而言，随机生成1至4之间一个自然数作为交叉位，每对染色体交换交叉位及之后基因信息（例如交叉位为3，则交换这对染色体3号位以及之后的基因信息）：

C_1'和C_2'交叉位为3，交换这对染色体3号位以及之后基因信息，得到新的染色体：

$$C_1'' = (1.0152, -3.9811, -0.1503, 0.0023)$$

$$C_2'' = (4.0589, 2.1904, -2.6638, 3.7535)$$

C_4'和C_5'交叉位为4，交换这对染色体4号位以及之后基因信息，得到新的染色体：

$$C_4'' = (1.0152, -3.9811, -2.6638, 1.8794)$$

$$C_5'' = (0.2073, 2.9932, -4.0802, 3.7535)$$

演化算法举例

步骤5：变异

假设变异概率为0.1。对于每个染色体中若干基因随机生成[0,1]的随机数，若该随机数小于0.1，则改变该基因信息取值，否则不改变该基因信息取值：

C_3''的一号位基因采样所得随机数小于0.1，该基因发生随机变异，其它基因信息保持不变：

$$C_3'' = (1.0152, -3.9811, -2.6638, 3.7535) \rightarrow (3.0953, -3.9811, -2.6638, 3.7535)$$

C_4''的二号位基因采样所得随机数小于0.1，该基因发生随机变异，其它基因信息保持不变：

$$C_4'' = (1.0152, -3.9811, -2.6638, 1.8794) \rightarrow (1.0152, 0.0153, -2.6638, 1.8794)$$

演化算法举例

步骤6：重新评价染色体适应值，更新*Best*

步骤7：判断结束

如果符合算法终止条件，则输出所找到最优解*Best*，退出程序，否则返回步骤3继续执行。

演化算法应用案例

- 已知 100 个目标的经度、纬度如表所示。
- 我方有一个基地，经度和纬度为（70,40）。假设我方飞机的速度为 1000 公里 / 小时。我方派一架飞机从基地出发，侦察完所有目标，再返回原来的基地。在每一目标点的侦察时间不计，求该架飞机所花费的时间（假设我方飞机巡航时间可以充分长）。
- 用遗传算法研究上述问题

求解的遗传算法的参数设定如下种群大小$M = 50$；最大代数$G = 1000$；交叉率$p_c = 1$，交叉概率为1能保证种群的充分进化；变异率$p_m = 0.1$，一般而言，变异发生的可能性较小。

（1）编码策略

采用十进制编码，用随机数列$\omega_1\omega_2 \ldots \omega_{102}$作为染色体，其中$0 \le \omega_i \le 1$（$i = 2,3,\cdots,101$），$\omega_1 = 0$，$\omega_{102} = 1$；每一个随机序列都和种群中的一个个体相对应，例如9目标问题的一个染色体为

[0.23，0.82，0.45，0.74，0.87，0.11，0.56，0.69，0.78]

其中编码位置i代表目标i，位置i的随机数表示目标i在巡回中的顺序，将这些随机数按升序排列得到如下巡回

6－1－3－7－8－4－9－2－5.

演化算法应用案例

（2）初始种群

先利用经典的近似算法—改良圈算法求得一个较好的初始种群。

对于随机产生的初始圈

$$C = \pi_1 \cdots \pi_{u-1}\pi_u\pi_{u+1} \cdots \pi_{v-1}\pi_v\pi_{v+1} \cdots \pi_{102}$$

$$2 \le u < v \le 101, \qquad 2 \le \pi_u < \pi_v \le 101$$

交换u与v之间的顺序，此时的新路径为

$$\pi_1 \cdots \pi_{u-1}\pi_v\pi_{v-1} \cdots \pi_{u+1}\pi_u\pi_{v+1} \cdots \pi_{102}$$

记

$$\Delta f = (d_{\pi_{u-1}\pi_v} + d_{\pi_u\pi_{v+1}}) - (d_{\pi_{u-1}\pi_u} + d_{\pi_v\pi_{v+1}})$$

若$\Delta f < 0$，则以新路经修改旧路经，直到不能修改为止，就得到一个比较好的可行解。

直到产生M个可行解，并把这M个可行解转换成染色体编码。

（3）目标函数

目标函数为侦察所有目标的路径长度，适应度函数就取为目标函数。我们要求

$$min\, f(\pi_1, \pi_2, \cdots, \pi_{102}) = \sum_{i=1}^{101} d_{\pi_i\pi_{i+1}}$$

演化算法应用案例

（4）交叉操作

交叉操作采用单点交叉。设计如下，对于选定的两个父代个体$f_1 = \omega_1\omega_2 \ldots \omega_{102}$，$f_2 = \omega'_1\omega'_2 \ldots \omega'_{102}$，我们随机地选取第$t$个基因处为交叉点，则经过交叉运算后得到的子代个体为s_1和s_2，s_1的基因由f_1的前t个基因和f_2的后$102-t$个基因构成，s_2的基因由f_2的前t个基因和f_1的后$102-t$个基因构成，

例如：

$$f_1 = [0,\ 0.14,\ 0.25,\ 0.27,\ |\ 0.29,\ 0.54, \cdots, 0.19,\ 1]$$
$$f_2 = [0,\ 0.23,\ 0.44,\ 0.56,\ |\ 0.74,\ 0.21, \cdots, 0.24,\ 1]$$

设交叉点为第四个基因处，则

$$s_1 = [0,\ 0.14,\ 0.25,\ 0.27,\ |\ 0.74,\ 0.21, \cdots, 0.24,\ 1]$$
$$s_2 = [0,\ 0.23,\ 0.44,\ 0.56,\ |\ 0.29,\ 0.54, \cdots, 0.19,\ 1]$$

交叉操作的方式有很多种选择，应该尽可能选取好的交叉方式，保证子代能继承父代的优良特性。同时这里的交叉操作也蕴含了变异操作。

（5）变异操作

变异也是实现群体多样性的一种手段，同时也是全局寻优的保证。具体设计如下，按照给定的变异率，对选定变异的个体，随机地取三个整数，满足$1 < u < v < w < 102$，把u, v之间（包括u和v）的基因段插到w后面。

（6）选择

采用确定性的选择策略，也就是说在父代种群和子代种群中选择目标函数值最小的M个个体进化到下一代，这样可以保证父代的优良特性被保存下来。

THANKS !

QUESTION?