Exploring Multi-dimensional Data via Subset Embedding (PDF)

1.简介

本文提出了一种 **多维数据分析方法**，目标是找到多维数据中包含的数据模式。该方法包含两部分：

- 一种基于神经网络的多维数据表征模型
- 一套基于该表征模型的可视化系统

2. 分析任务

多维数据，如图1所示 (Chicago Crime 数据集)

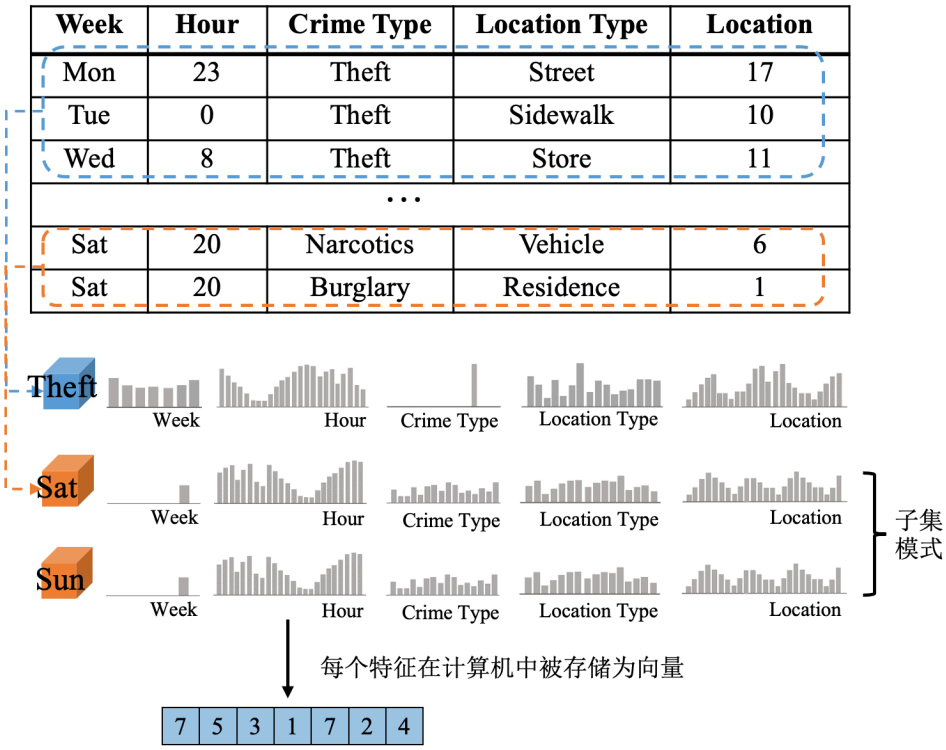

图 1

每行表示一条犯罪记录，包含5个属性，整个数据集共有约6百万条数据。

定义：

- **子集**：包含任意记录的集合。如图1中蓝色立方体表示“Theft子集”，它包含“Crime Type”属性为“Theft”的全部犯罪记录；橙色立方体表示“Saturday子集”，它包含“Week”属性为“Saturday”的全部犯罪记录。
- **子集的特征**：特征是指子集包含的数据记录的一种**统计信息**。如图1，蓝色的Theft子集有5个特征（使用灰色直方图表示），分别对应数据集的五个属性。每个特征在计算机中被存储为一个向量。
- **子集的模式**：子集模式是指具有相似特征的一组子集。如图1中橙色的两个子集具有相似的特征，可被视作一种子集模式。
- **分析任务**：挖掘多维数据集中存在的子集模式。

3. 现有方法的问题

- **聚类分析**：挖掘数据记录之间的关系 (行关系)
- **子空间分析**：挖掘不同特征之间的关系 (列关系)

然而，**子集模式往往同时与记录和特征相关。**例如图2中所示，两个绿色子集仅在部分特征上 (黄色标记) 相似，这是现有方法无法处理的。

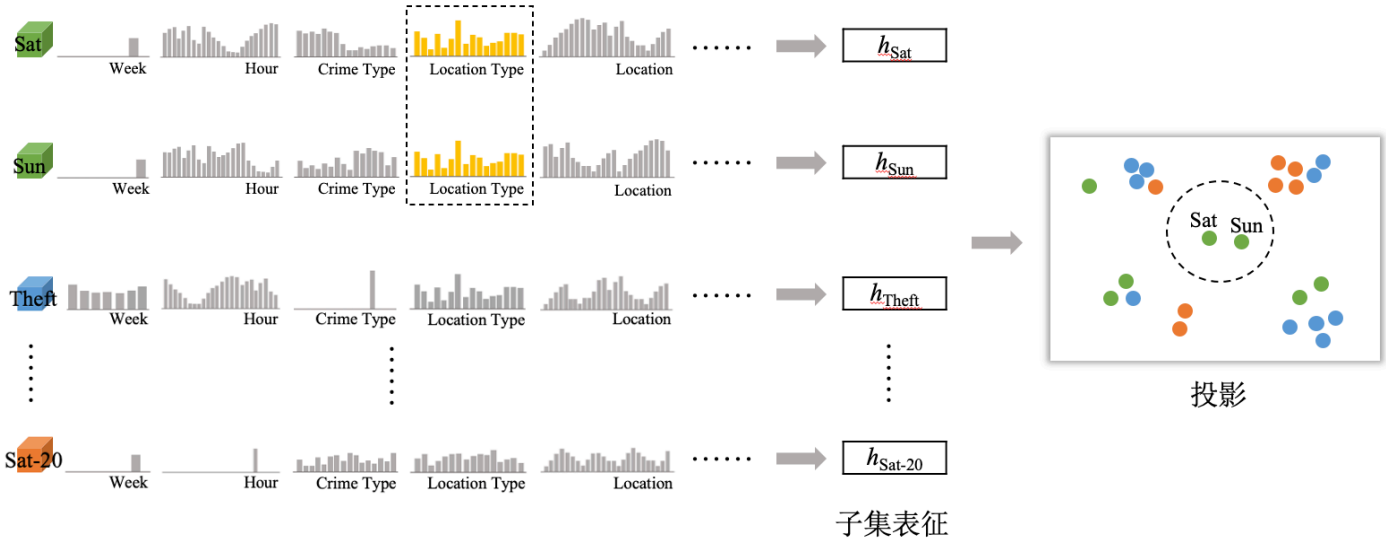

图 2

4. 寻找子集模式的难点

子集数量很**多**，每个子集的特征也很**多**，如果人工地去比较每一对子集，搜索空间是巨大的。

我们使用的方法是，**为每个子集生成统一格式的表征向量**，来表示子集的全部特征。再将子集投影到散点图，如图2所示。投影中靠近的点，表示这两个子集具有相似的特征，从而可以快速找到子集模式。

5. 子集表征网络 (SEN)

5.1 需求

1. 表征**具有不同特征的子集** (异构)

图 3

2. **当子集仅具有少量相似特征时，也能够为它们生成相似的表征**

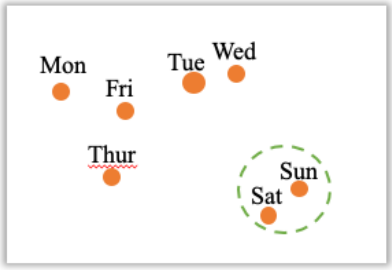

图 4

5.2 模型设计

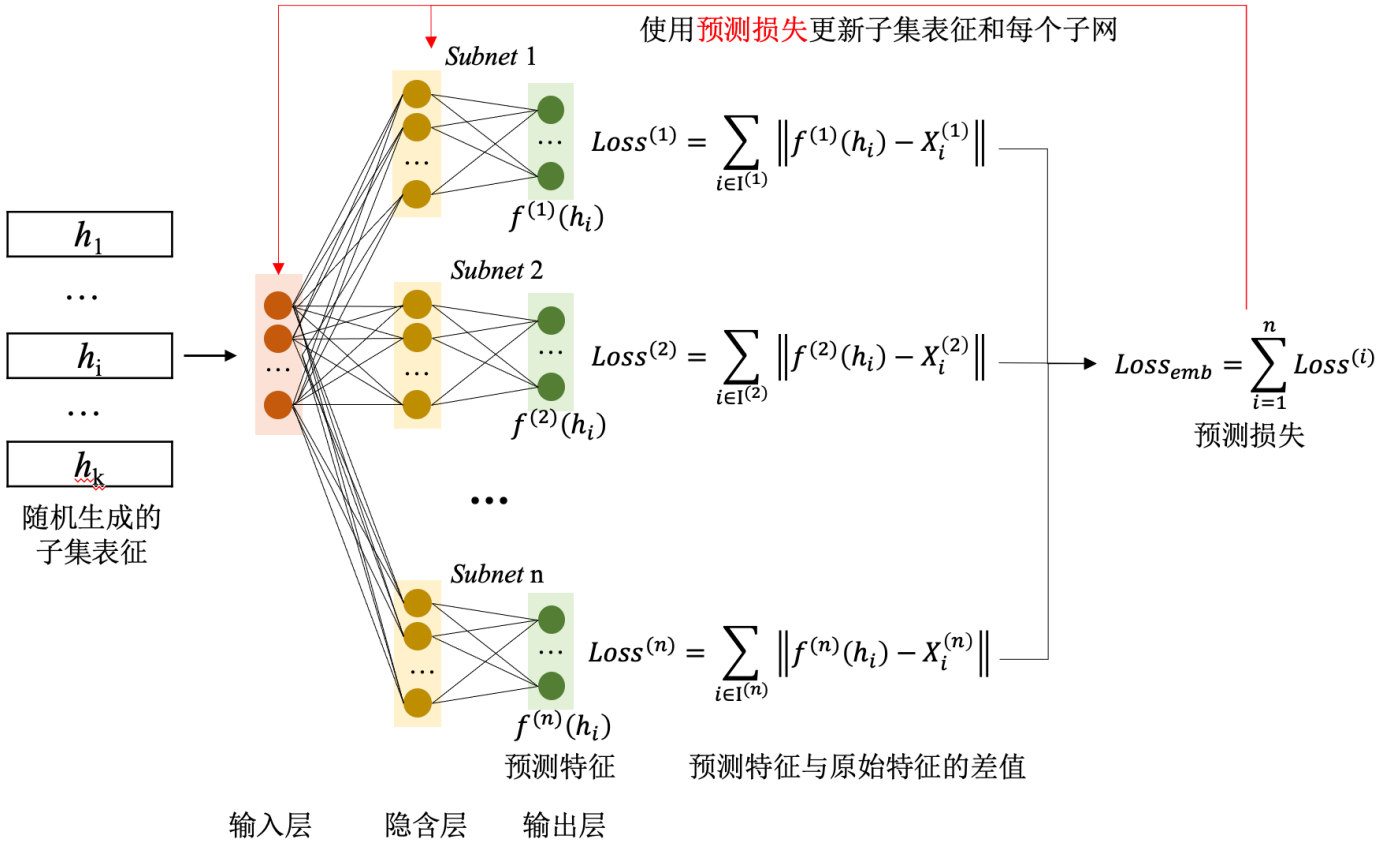

图 5

我们提出了一种子集表征模型，如图5所示。它由多个子网组成，每个子网是包含一个隐含层的全连接神经网络，对应一个特征。

训练前：

1. 随机初始化**每个子集的表征向量**
2. 随机初始化**每个子网的参数**

前向传播：

子集的表征**只传入它具有的特征对应的子网**。如图6所示，子集1的表征只会被传入子网1和子网n。这样可以处理子集具有不同特征的情况 (**需求1**)。

子网输出是对应的预测特征。

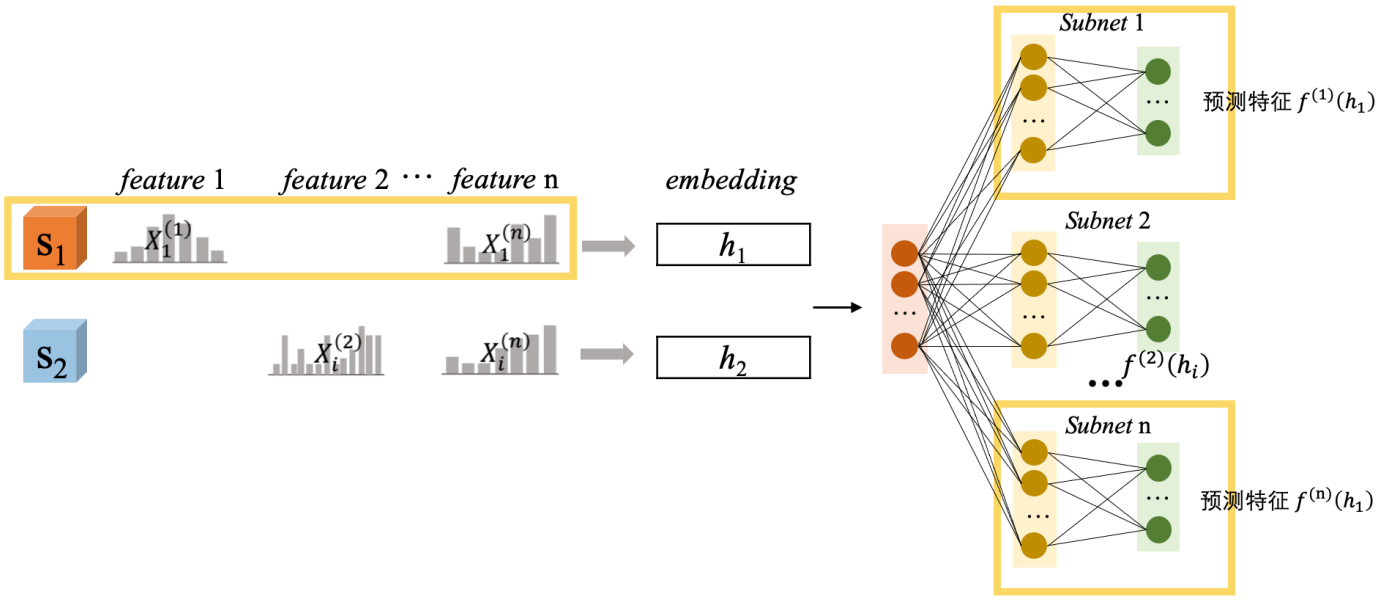

图 6

反向传播:

计算全部特征的预测损失 (预测特征与真实特征之差)，如图5所示，使用预测损失更新子集表征和子网参数。

总结:

这样的模型设计类似于网络信号传输：对于从远处传过来的信号，经过不同解码器 (子网) 可以得到不同的信息 (特征)。

5.3 损失函数推导

问题：已知子集S具有N个特征$\{X^{(1)},\ldots,X^{(N)}\}$，求它的表征$h$。

模型输入是未知的表征h，输出是已知的子集特征。可以利用极大似然法，通过已知的结果信息，反推最有可能的导致这种结果出现的输入。

子集表征h的似然函数为：

$$L(h|S)=P(S|h)=\prod_{n=1}^{N}P(X^{(n)}|h) \tag{1}$$

我们将给定子集表征h时输出为$X^{(n)}$的概率建模为

$$P(X^{(n)}|h)\propto e^{-\delta(X^{(n)}-f(h,\theta^{(n)}))} \tag{2}$$

其中$\theta^{(n)}$表示子网n的参数。为了便于计算，我们在式子两边取对数，即

$$log(L(h|S)) \propto -\sum_{n=1}^{N} \delta(X^{(n)} - f(h, \theta^{(n)})) \tag{3}$$

我们训练模型的目标是使得$log(L(h|S))$尽量大，即使得$\sum_{n=1}^{N} \delta(X^{(n)} - f(h, \theta^{(n)}))$尽量小。从而引出我们的损失函数，对于一个具有K个样本的训练集，损失函数是

$$Loss_R = \frac{1}{K} \sum_{n=1}^{N} \sum_{k \in I^{(n)}} ||f(h_k, \theta^{(n)}) - X_k^{(n)}|| \tag{4}$$

其中$I^{(n)}$表示具有特征n的样本集合，称 $Loss_R$ 为预测损失。

5.4 模型训练

训练过程伪代码

Algorithm 1: Subset Embedding Network Training

Input: subsets $\{S_1, ..., S_k\}$, $S_i = \left\{X_i^{(1)}, ..., X_i^{(n)}\right\}$

Randomly initialize embeddings $\{h_1, ..., h_k\}$ and parameters of subnets $\left\{\theta^{(1)}, ..., \theta^{(n)}\right\}$

while *not converged* **do**

  Input $\{h_1, ..., h_k\}$ to the corresponding subnets.

  Calculate $Loss_R = \frac{1}{K} \sum_{n=1}^{N} \sum_{k \in I^{(n)}} ||f(h_k, \theta^{(n)}) - X_k^{(n)}||$

  for *n = 1 : N* **do**

    Update the network parameters $\theta^{(n)}$ with gradient descent:

    $\theta^{(n)} \leftarrow \theta^{(n)} - \alpha \cdot \partial Loss_R / \partial \theta^{(n)}$

  end

  for *k = 1 : K* **do**

    Update the representation h_k with gradient descent:

    $h_k \leftarrow h_k - \alpha \cdot \partial Loss_R / \partial h_k$

  end

end

Output: embeddings $\{h_1, ..., h_k\}$

代码实现：TensorFlow 1.14

细节：

- 特征预处理

  对每个Field的特征向量进行归一化处理：$X = \frac{X - \mu}{X_{max} - X_{min}}$

- Epoch

使用全部子集训练，由于子集数量较少，因此不再分batch。
Epoch设为50。训练开始时，损失下降很快，后面逐渐减小，接近50轮时，损失依然在下降，但是下降幅度很小。由于表征模型要集成到数据可视化系统中，为了不影响系统响应速度，我们设Epoch=50。

5.5 实验

验证：当子集仅具有少量相似特征时，能否为它们生成相似的表征（**需求2**）。

1. **数据集**

   Handwritten Digits数据集包含2000个样本，每条样本有6个特征 (每个特征是100维左右的向量)，包括像素信息，字符轮廓相关性等。数据集共包含10个类别。

2. **对比方法**

   t-SNE，m-SNE

3. **实验设计**

   原始数据中同类样本具有相似的特征，我们使用随机数替换掉部分原始特征，使得同类对象只有部分特征相似，以模拟现实中子集只有少量相似特征。

   （1）实验分为6轮，每轮逐渐增多被随机数替换的特征数量。具体来说，第一轮使用全部原始特征，第二轮使用随机数替换掉一个特征，第三轮使用随机数替换两个特征，依次类推。每轮中分别使用三种技术为处理后的数据集生成表征。

   （2）使用KMeans对三种方法产生的表征做聚类，将聚类结果与真实标签进行对比，使用三个评价指标ACC，NMI，ARI，每个指标都是越大越好。

   （3）投影三种方法产生的表征，比较每个类别包含的样本在投影中的紧凑性，使用两个评价指标SC，CHI，每个指标越大越好。

4. **实验结果**

   随着被随机数替换的特征数量增多，所有指标值都有所下降，但是我们的方法在各个指标上下降得最慢，如图7所示。

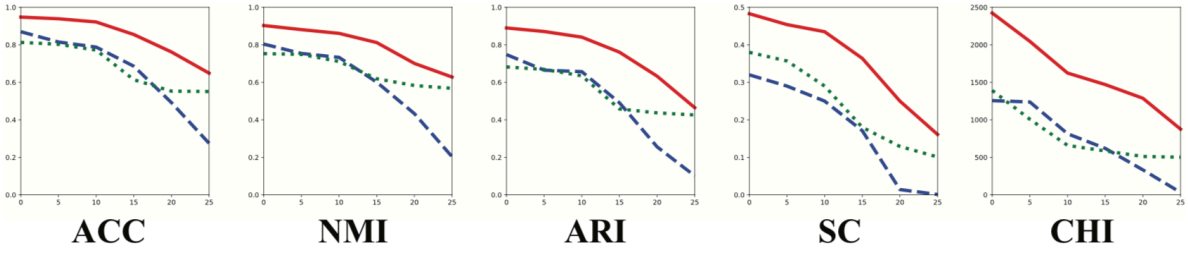

图 7

我们为每一轮中三种技术产生的子集表征生成了投影 (散点图)，图中每个点表示一个原始记录，颜色表示类别。可以发现，当仅有少数特征被随机数替换时，在三种技术产生的投影中都可以清晰地识别出每个类别 (图8左侧两列)；但随着被随机数替换的特征数量增多 (从左向右)，在其他两种技术生成的投影中，各种类别的样本混合在一起，而我们的方法依然可以清晰地展示各个类别的样本。

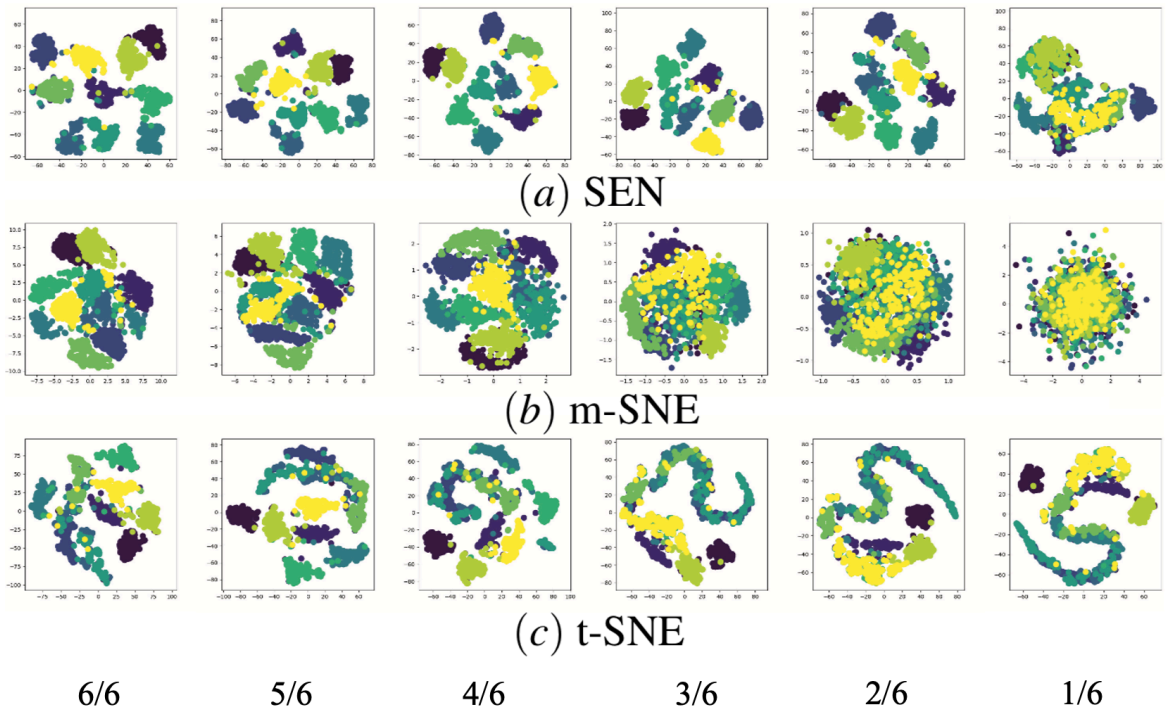

图 8

5. **分析**

   我们提出的子集表征模型表现最佳的原因是，特征之间没有交叉，**每个特征都与子集表征建立了直接的映射关系**。

   如图9所示，如果先把全部特征拼接为一个向量，只建立一个预测网络，则会引起特征交叉。假设两个样本的feature1相似，而feature2不相似，由于特征之间存在交叉，无法为这两个样本生成相似的表征，就像是feature2把feature1污染了一样。

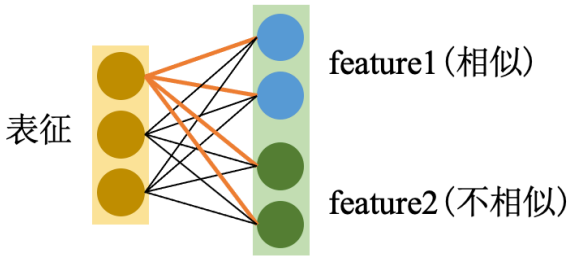

图 9

6. 可视化系统设计

我们设计了一个用于探索子集模式的可视化系统，系统中包含三个组件 (a-c)，如图10所示。

图 10

(a) 切分数据集为子集 (树形图)

我们提出了一种渐进式的数据集切分策略：

1. 在每一轮只允许选择一个属性划分子集
2. 前一轮生成的子集在下一轮探索时还可以再次被切分，以探索更细致的数据模式。

(b) 表征子集，生成散点图

利用表征模型为每个子集生成表征向量，该向量编码了子集的全部特征。之后将该向量投影到二维平面(散点图)，在散点图中每个点表示一个子集，子集的距离表示它们特征之间的相似性，距离越近，表示两个子集具有的相似特征越多。

(c) 可视化数据模式 (折线图等)

用户可以在散点图中选择距离近的一组子集，并利用特征卡片观察他们具体的特征值。

7. 可视化系统实现

7.1 Data Cube

我们需要为生成的子集提取特征，特征是子集包含的记录的数量在每个属性上的分布，这设计到大量的Count操作。要分析的数据集包含6百万条记录，如果使用MySQL存储，考虑一种可视化系统中常见的场景：为7个Week子集，每个生成4个特征向量（四个向量长度总和为24+16+15+24=79），需要对整个数据集执行7*79=553次Count操作，这是很耗时的，会影响可视化系统的响应速度。

我们构建了Data Cube (图 11)，使用预计算的方式加速计算Count值。

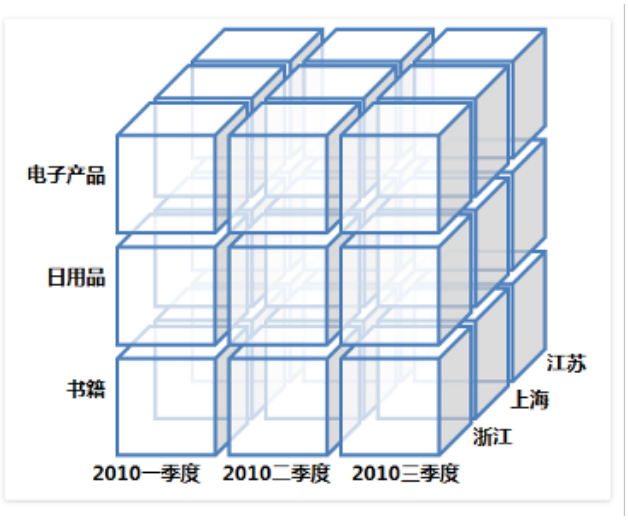

图 11

1. Chicago Crime数据集的五个属性都是离散型，全部属性值有$7 * 24 * 16 * 15 * 24 = 967,680$种组合，分别计算这些组合对应的记录数量，存储在csv文件中。
2. 用Python将Cube数据转化为Dict数据结构存储在内存中，如图12所示。

ID	Week	Hour	Type	Location	Map	Count
0	1	0	Theft	Store	0	300
1	1	1	Theft	Store	0	1002
			.....			
967679	7	23	Robbery	School	24	500

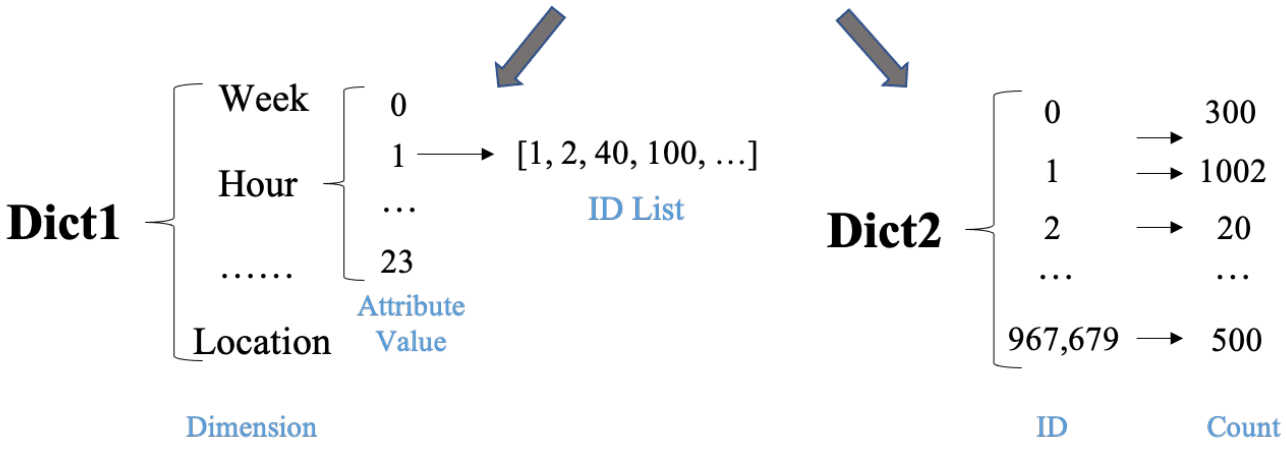

图 12

3. 读取数据

   例如，查询周末早上10-12点的子集{Week:[6,7],Hour:[10,11,12]}。在Dict1中找到Week=[6,7]的对应的id列表求并集，找到Hour=[10,11,12]的id列表求并集，再将两个id列表求交集。依照得到的交集列表，到Dict2中找值求sum

7.2 Web系统实现

1. 后端

   使用Python Flask框架，主要包含以下模块：

   - 生成投影：调用Data Cube读取数据；调用模型表征子集
   - 探索投影：查询子集；对投影中的子集进行聚类，使用SKLearn库KMeans算法

2. 前端

   使用Vue.js框架。树状图、散点图、折线图等绘制使用D3.js完成。

8. 可视化系统演示

8.1 可视化系统操作演示

8.2 数据分析案例