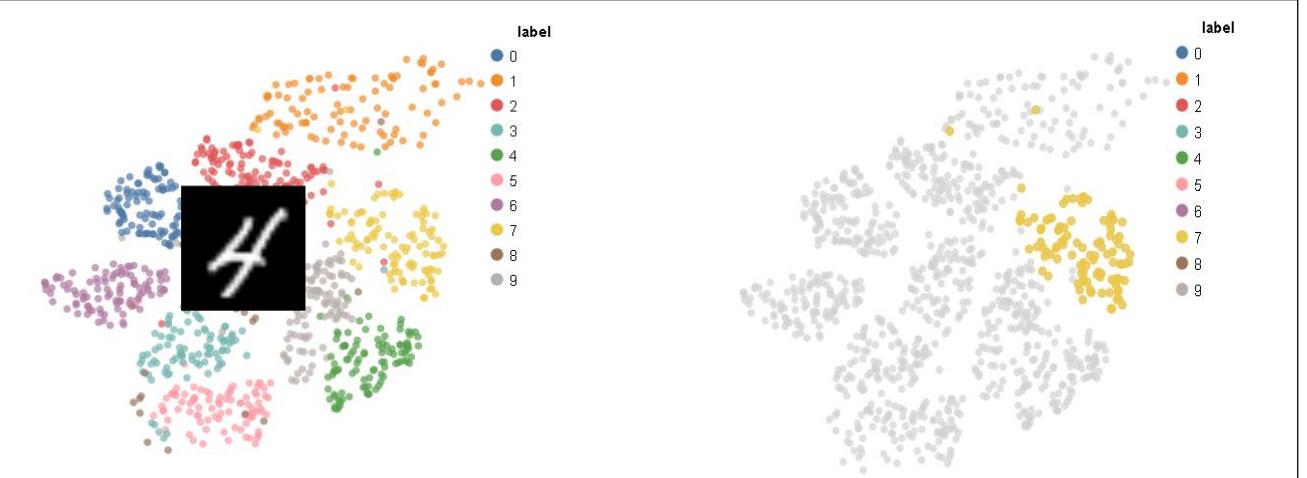


# 山东大学 计算机科学与技术 学院

## 大数据分析实践 课程实验报告

学号: 202300130236	姓名: 陈德康	班级: 数据 23		
实验题目: Canis/Cast/Libra 实践				
实验学时: 2	实验日期: 2025.11.14			
实验目的: 从 Canis/Cast/Libra 这三个工具中, 任选其一完成实践				
硬件环境: 计算机一台				
软件环境: Windows 11				
<b>实验步骤与内容:</b> <b>1. Libra 简介</b> 在 Libra 中, 作者提出了一套可视化交互模型解决现有可视化库交互难以复用、拓展和组合的问题: - D3 仅提供少量预定义交互 (刷选、拖拽、缩放), 自定义交互需手动管理状态, 难以复用扩展; - Vega 依赖函数式响应式编程, 交互与视觉表示强耦合, 适配其他可视化场景困难; - 多数库缺乏撤销/重做、交互组件模块化等关键能力, 新交互技术难以推广  为了解决这些问题, Libra 库以模块化、可复用、兼容现有库为目标的该方案, 核心组件包含图层、工具与交互效果, 其中图层通过将可视化与交互元素分离为背景层、主层、选择层、临时层以避免交互干扰可视化本身, 工具作为用户与数据的中介整合了将低级别事件转为高级动作的交互器和封装选择、布局、分析等核心操作的服务, 交互效果则涵盖提前展示操作结果的前馈、操作后视觉响应的反馈以及支持撤销/重做的命令  <b>2. 实验内容</b> 在官方给出的示例网站中, 有使用 Libra 实现了 MINST Hover 和 MNIST Click 的功能, Hover 功能是让鼠标悬停的样本点展示其图片, Click 的功能是鼠标长按点击高亮该点所属类别的所有点, 效果分别如下:				



那么可以尝试直接在网页源代码中的 index，整合这两个功能，即默认图片灰色，鼠标悬停展示样本点图片，长按高亮该点所属类别的所有样本点，代码如下：

```
// import static visualization and global variables
const VIS = require("./staticVisualization");

async function main() {
  await VIS.loadData();
  VIS.renderStaticVisualization();
  const mainLayer = renderMainVisualization();
  mountInteraction(mainLayer);
}

function renderMainVisualization() {
  // Find the SVG element on page
  const svg = d3.select("#LibraPlayground svg");
  // Create the main layer
  const mainLayer = Libra.Layer.initialize("D3Layer", {
    name: "mainLayer",
    width: globalThis.WIDTH,
    height: globalThis.HEIGHT,
    offset: { x: globalThis.MARGIN.left, y: globalThis.MARGIN.top },
    container: svg.node(),
  });
  const g = d3.select(mainLayer.getGraphic());
  // Draw points code
  g.selectAll("circle")
    .data(globalThis.data)
    .join("circle")
    .attr("class", "mark")
    .attr("cx", (d) => globalThis.x(d[globalThis.FIELD_X]))
    .attr("cy", (d) => globalThis.y(d[globalThis.FIELD_Y]))
}
```

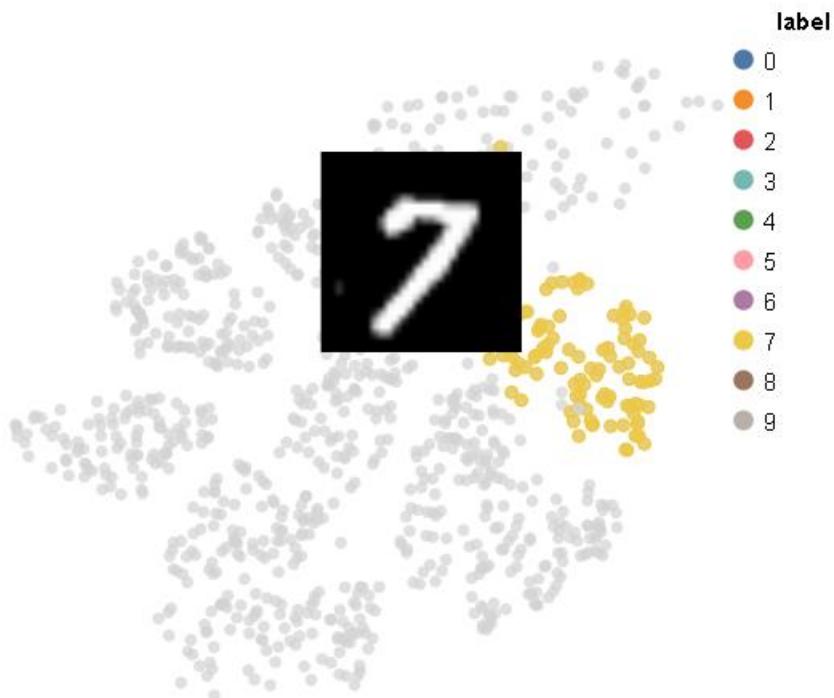
```
.attr("fill", "lightgray")
.attr("fill-opacity", 0.7)
.attr("r", 3);
return mainLayer;
}

async function mountInteraction(layer) {
// 点击高亮功能
Libra.Interaction.build({
inherit: "ClickInstrument",
layers: [layer],
remove: [
{
find: "SelectionTransformer",
},
],
insert: [
{
find: "SelectionService",
flow: [
{
comp: "FilterService",
sharedVar: {
fields: ["label"],
},
},
{
comp: "SelectionTransformer",
layer: layer.getLayerFromQueue("selectionLayer"),
},
],
},
],
sharedVar: {
highlightColor: (d) => globalThis.color(d[globalThis.FIELD_COLOR]),
},
});
// 悬停展示图片功能
Libra.Interaction.build({
inherit: "HoverInstrument",
layers: [layer],
sharedVar: {
tooltip: {
image: (d) => d.image, // 悬停时展示数据中的image字段
}
}
});
```

```
        offset: {
          x: -70 - globalThis.MARGIN.left,
          y: -100 - globalThis.MARGIN.top,
        },
      },
    },
  });
  await Libra.createHistoryTrrack();
}

main();
```

效果如下图：



#### 结论分析与体会：

通过本次实验，我了解了 Libra 库的设计动机和核心优势，并在官方示例的网页中使用 Libra 库整合了 MINST Hover 和 MNIST Click 的功能，实现了悬停展示图片功能的同时高亮长按点击的点所属类别的所有样本点，体会到了 Libra 库设计的优势，这为我进行进一步的学习和进行更复杂的实验奠定基础