作业选择**TSNE-数据可视化降维**和**Squarified Treemaps 算法实现**
源代码文件在相应的文件夹下

TSNE-数据可视化降维

一、运行

```
python tsne.py
```

二、算法原理

2.1 SNE原理

SNE即stochastic neighbor embedding，其基本思想为在高维空间相似的数据点，映射到低维空间距离也是相似的。SNE把这种距离关系转换为一种条件概率来表示相似性。

假设高维空间中有 x_i, x_j 两个点，$p_{j|i}$ 表示中心为 x_i 时，x_j是其近邻点的概率：x_j 越靠近 x_i其值越大，反之概率就越小。$p_{j|i}$采用高斯分布，公式如下：

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

对于不同的中心x_i，其对应的高斯分布的方差σ也不同,需要对每个点进行计算。

同样对于高维空间的点x_i, x_j映射为低维空间对应的点为y_i, y_j，其概率分布函数$q_{j|i}$如下。在这里为了方便计算，假设所有点的σ都为$\frac{1}{\sqrt{2}}$

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k\neq i} exp(-||y_i - y_k||^2)}$$

为了让高维空间的点映射到低维空间后，尽可能保持一样的分布，即原来离得近的点还离得近，离得远的点还离得远，所以要保证两个分布尽可能相似，这里用的衡量的方法就是采用KL(Kullback-Leibler Divergence)距离。

$$C=\sum_i KL(P_i||Q_i)=\sum_i\sum_j p_{j|i}log\frac{p_{j|i}}{q_{j|i}}$$

现在问题转化了如何让上面的代价函数C最小。经典方法就是**梯度下降法。**

$$\frac{\delta C}{\delta y_i}=2\sum_j(p_{j|i}-q_{j|i}+p_{i|j}-q_{i|j})(y_i-y_j)$$

在进行梯度更新时，要注意Cost Function并不是一个凸函数，所以会存在很多的局部最优解，为了防止陷于局部最优解，还需要加上一个“动量”，可以冲出局部最优解。

$$Y^t=Y^{t-1}+\eta\frac{\delta C}{\delta y_i}+\alpha(t)(Y^{t-1}-Y^{t-2})$$

其中Y^t表示t次迭代的结果，η是学习率，$\alpha(t)$表示第t次迭代时的动量。

此时还剩下一个问题没有解决：如何为不同的x_i选择合适的σ。这里提出叫做困惑度(perplexity)的概念，来表示x_i附近的有效近邻点的个数，通常取5-50之间。

$$Perp(P_i)=2^{H(P_i)}$$

$$H(P_i)=-\sum_j p_{j|i}log_2 p_{j|i}$$

给定困惑度之后，使用二分搜索寻找一个最合适的σ。

但是需要注意一点时，**KL距离具有不对称性。**

1. $p_{j|i}$越大，$q_{j|i}$越小时，此时的Cost越高。即高维空间的点越接近，映射到低维空间时反而越远，此时的惩罚是很大的，这是正确的情况。
2. $p_{j|i}$越小，$q_{j|i}$越大时，此时的Cost越小。即高维空间的点距离远时，映射到低维空间的点接近时，此时的惩罚却很小，这时跟我们想要的结果正好相反。

因此**SNE倾向于保留局部特征**，即让高维离得近的点尽可能在低维时聚在一起，但是不考虑不同类间的间隔，直观理解为：整个降维后的图像会比较“拥挤”。

2.2 t-SNE原理

t-SNE是在SNE的基础上进行了以下两点改进：

- 使用对称SNE，简化梯度公式
- 低维空间使用t分布取代高斯分布

我们先看改进1，将非对称的SNE改为对称的SNE。

在之前的条件分布概率中，是不对称的，例如高维空间中$p_{i|j}$是不等于$p_{j|i}$的，这与我们的直觉不符合，因为无论是x_i还是x_j谁作为中心点，其出现在对方附近的概率应该是相等的，所以我们应该设计一个联合概率分布，使得$p_{ij} = p_{ji}$.

于是在高维、低维空间中，我们重新定义一下概率分布，注意除号下面部分与之前的区别：

$$p_{ij} = \frac{exp(-||x_i - x_j||^2)}{\sum_{k\neq l} exp(-||x_k - x_l||^2)}$$

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k\neq l} exp(-||y_k - y_l||^2)}$$

对于高维空间中的点，为了避免异常值的影响，采取以下方法定义高维空间的联合分布：

$$p_{ij} = \frac{p_{i|j} + p_{i|j}}{2n}$$

此时KL距离组成的损失函数如下：

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{P_{ij}}{q_{ij}}$$

梯度为：

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

下面继续看t-SNE的第二个改进：低维空间用t分布替换高斯分布。这样做的好处是在低维的情况下，将同类的数据的距离减少，不同类间的距离拉大，这样可视化的效果会更好。

所以低维空间上的分布函数如下：

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k\neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

此时梯度如下：

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ji} - q_{ji})(y_i - y_j)(1 + |y_i - y_j||^2)^{-1}$$

三、算法实现

输入的数据为MNIST数据集中，抽取的2500条数据，每一条数据是784维，所以输入的规模为2500*784。

为了降低TSNE执行的复杂度，在进行TSNE之前，先通过PCA对数据进行降维，减少参数量，简化TSNE计算，否则实际的执行时间过长。

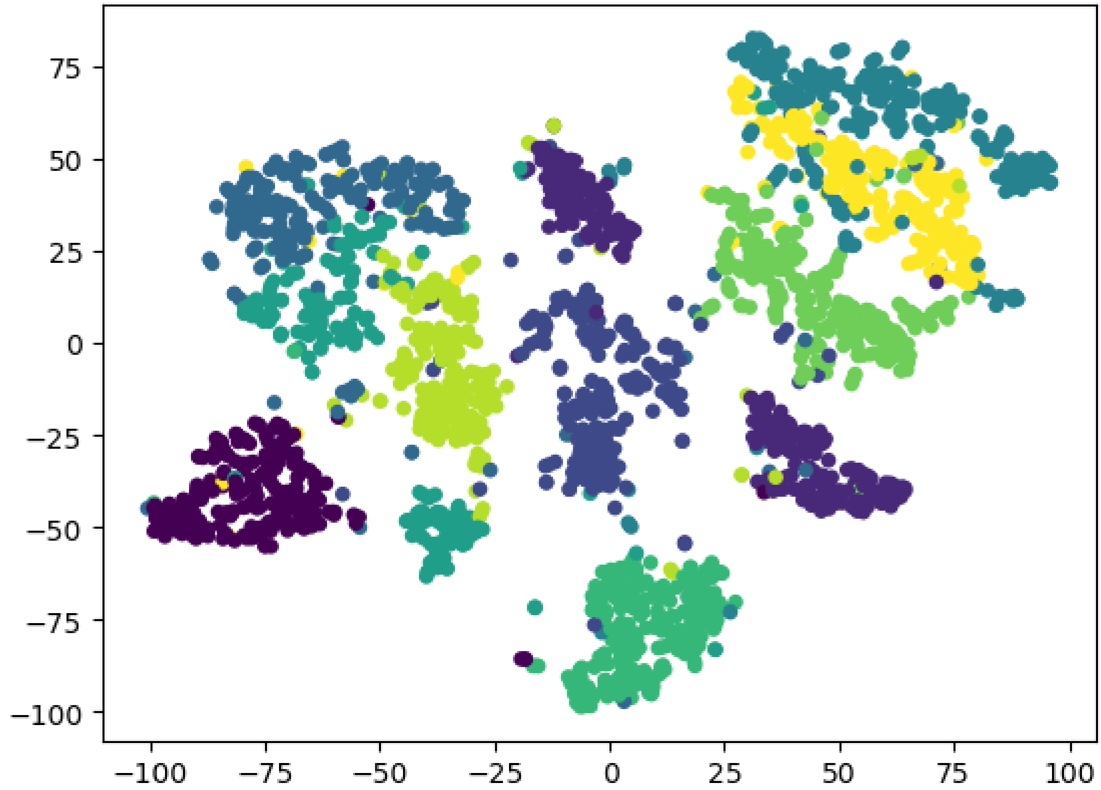
算法伪代码如下：

```
input data set X = {x1, x2, ..., xn}
input perplexity Perp
input iterations T, learning rate n, momentum a(t)

begin
    compute p_{j|i} with perplexity Perp
    compute P_{ij}
    initial y(0) = {y1, y2, ..., yn}

    for t = 1 to T
        compute q_{ij}
        compute gradient
        update y(t)
    end
end
```

四、实验结果



五、参考文献

1. Van der Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of Machine Learning Research, 2008, 9(2579-2605): 85.

Squarified Treemaps 算法实现

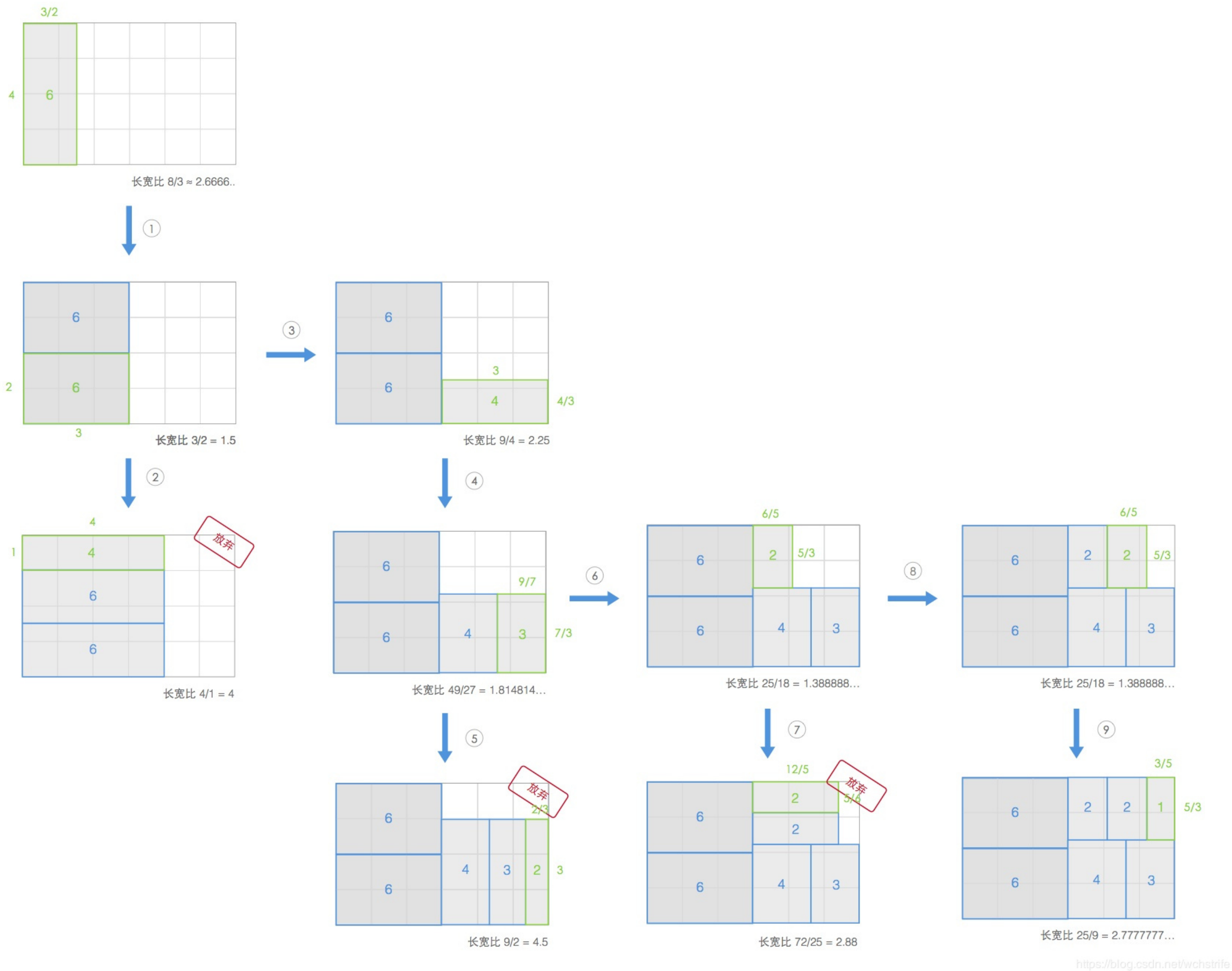
- 实现Squarified Treemaps算法
- 对treemap进行可视化
- 可以自定义深度和宽度
- 自动生成数据
- 过渡动画

一、 算法原理

主要的工作是复现了"Squarified Treemaps"中的算法。

算法的总体思想是使生成的图形接近正方形，在每一步计算当中计算最新放置矩形的长宽比，长宽比越接近1则越接近正方形，当长宽比偏离1时，应放弃当前插入的位置，重新寻找下一个插入的位置。

在这里借用知乎上的一张图片很清晰的展示了算法执行的过程。



利用递归的思想，按照水平和垂直的方向排列矩形，排列时根据是否会改善当前的长宽比，来决定要么将矩形添加到当前行，要么固定当前行，在剩余的子矩形中添加新的行。

程序的伪代码如下：

```
procedure squarify(list of real children, list of real row, real w)
begin
    real c = head(children);
    if worst(row, w) <= worst(row++[c], w) then
        squarify(tail(children), row++[c], w)
    else
```

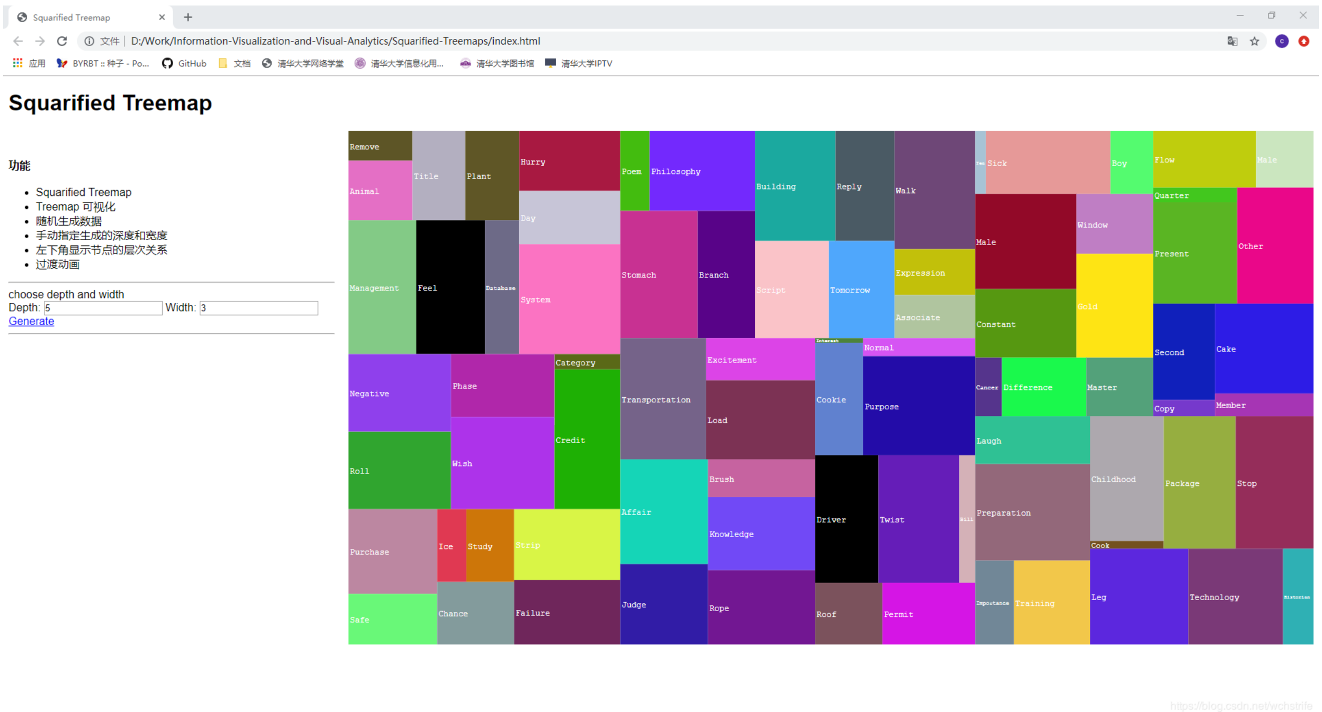
```
        layoutrow(row);
        squarify(children, [], width());
end
```

其中,width()返回给出当前所在行的其余子矩形的最短边的长度。

layoutrow()在矩形中添加新的子行。

worst()返回矩形列表中的最高长宽比。

二、实验效果



三、参考资料

1. Bruls, Mark, Kees Huizing, and Jarke J. Van Wijk. Squarified treemaps." In Data Visualization 2000, pp. 33-42. Springer, Vienna, 2000.
2. 来，认识一下这个数据可视化中的90后：Treemap - Xhinking的文章 - 知乎 https://zhuanlan.zhihu.com/p/19894525