计算机应用数学课程论文

范逍乾(11521021)

May 31, 2016

1 引言

T-SNE算法是一种用于可视化高维数据的算法。具体来说，t-SNE算法是一种降维算法，它能够将高维数据点转化为低维数据点，比如二维或者三维的数据点，从而实现高维数据点在低维空间的可视化表示。目前高维数据可视化的方法有很多，但传统的高维数据可视化算法仅仅是对高维数据进行可视化展示，却不能给观察者提供较为直观的信息。因此考虑使用降维算法，通过将高维数据转化为能够在散点图上直观表示的2维或3维的数据，来实现对高维数据的可视化表示。降维算法的主要目标是要能保证降维后的低维数据能够最大程度地保留原有高维数据的结构信息。目前传统的降维算法有PCA，MDS，这类降维算法属于线性降维算法，在降维过程中只考虑尽可能分离不相似的点，而没有考虑相似数据点的聚集。为了保证相似的高维数据点在降维后能尽可能聚集到一起，考虑使用非线性的降维算法。其中，SNE就是一种非线性的降维算法，另外还有一些降维算法如，Sammon mapping，Isomap，Maximum Variance Unfolding，Locally Linear Embedding，Laplacian Eigenmaps等。然而上述这些非线性的算法在可视化一些现实的高维数据，如手写数字数据时，最终的可视化效果不是十分理想。主要问题是，这些算法都能较好地保留高维数据的局部结构，但是却没法同时保留高维数据的全局和局部结构。T-SNE算法是一种改进的SNE算法，相比原来的SNE算法，T-SNE算法能够在处理高维数据时得到更好的可视化效果。

2 方法概述

首先，先介绍SNE算法，SNE算法的全称Stochastic Neighbor Embedding，该算法的主要思路是计算高维数据点两两之间的相似度p，之后计算降维后相应的低维数据点之间的相似度q，之后计算高维数据点间相似度和降维后低维数据点间相似度的差异，并不断调整低维数据点，最终当两者差异达到最小时，得到低维数据点的最优结果。

具体来说，两个高维数据点的相似度的计算方法是，先计算两个数据点i，j之间的欧氏距离，之后将以数据点i为中心的高斯分布函数作为概率密度函数，计算出相应的概率值，最后计算出条件概率$p_{j|i}$，将其值作为两个点之间的相似度值。具体计算公式为：

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

同理，对于高维数据降维后对应的低维数据，也要计算其相应的相似度$q_{j|i}$，计算方法与高维数据点的计算方法相同，只是这里给定高斯分布的方差σ^2 为$\frac{1}{\sqrt{2}}$。

SNE算法通过Kullback-Leibler距离来表示$p_{j|i}$和$q_{j|i}$之间的差异，具体的成本函数：

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

SNE算法的目的是要获得一组低维数据点，使得所有的Kullback-Leibler距离之和，也就是成本函数的值尽可能小，这里使用梯度下降方法来实现这一目标。根据这个成本函数公式，可以看出，SNE算法能够很好地保留的原始数据中相似度高的数据点的结构信息，即距离近的数据点在降维后，仍然可以维持较近的距离，但会出现原始数据中相似度小的点，也就是距离较远的点，在降维后也会形成一定程度的聚集。所以，SNE算法只能较好地表示高维数据的局部结构。

这里还需要确定的是高维数据相应的高斯分布函数的方差值，不同的点i会对应不同的方差值。具体方差确定的方法：采用二分查找法确定，最终方差要能满足预先设定的perplexity值，涉及到的公式主要为：

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

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

成本函数对应的梯度公式是：

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

该梯度公式可以形象地理解为低维数据点i 与其他所有低维数据点j之间用弹簧连接起来，两个点之间的弹簧会产生相应的排斥力和吸引力。从公式中可以看出，当两个低维空间的数据点之间距离较远时，会有较大的吸引力，而当这两个点之间距离较近时，会产生一定的排斥力，或者说会产生较小的吸引力。在SNE算法中，低维数据的点最开始是通过一个基于原点的高斯分布函数，随机生成一系列相应的低维数据点，之后再通过梯度下降的方法对每个数据点进行调整，优化。

考虑到sne算法的优化过程相对比较复杂，同时还会出现“拥挤问题”，为了解决这两类问题，t-SNE算法在两方面做了调整：1.对高维数据点之间的相似度计算，采用对称SNE方法来进行计算。2.对低维数据点之间的相似度计算，采用Student-t分布进行计算，代替原有的高斯分布。

具体来说，在计算数据点之间的相似度值时，t-SNE算法用联合概率分布代替了SNE的条件概率分布，最终得到成本函数：

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

对于低维数据点之间的相似度计算，采用公式q_{ij}：

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

对于高维数据点，考虑到一些分布较“偏远”的数据点，它们所对应的p_{ij}值相对较小，从而导致它们在整个成本函数中占的比重非常小，这容易导致最终降维后的数据结果不准确。为了避免出现这种情况，考虑采用下面公式计算p_{ij}：

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

可以看出，此时，p_{ij}和p_{ji}的计算结果是一样的。同时，这也保证每个高维数据点都能够在成本函数中占到一定的比重。

采用这种对称的SNE算法最终得到的梯度公式也更加简单：

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

SNE算法还会出现“拥挤问题”。“拥挤问题”是指SNE在降维的过程中会出现原来两个相距较远的高维数据点在降维后，距离反而拉近的情况。这种情况通过观察成本函数就能得出。通过观察梯度函数，也可以发现，两个相距较远的数据也会有一定的吸引力，即便这两个数据点在高维空间中的距离相对较远。这样一来，由于原先距离较远的点也趋向于聚在一起，会导致最终降维后的数据的聚类效果一般，也就是所谓的“拥挤问题”。为了解决“拥挤问题”，一种方法是采用UNI-SNE算法，该算法会对qij值进行优化，但是其优化过程相对比较复杂。

为了解决上述的“拥挤问题”，t-sne算法使用了Student-t分布，代替原有的高斯分布对两个点之间的相似度值进行计算。使用Student-t分布计算q_{ij}的公式：

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

最终得到相应的梯度公式：

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

同原先的SNE相比，t-sne解决了“拥挤问题”，对于相距较远的高维数据点，其对应的低维空间的数据点有更大的排斥力，同时，与UNI-SNE相比，t-SNE不会提供非常大的排斥力，从而导致相应低维数据点分离过远。总的来说，t-SNE通过梯度下降的方式来优化成本函数，其具体流程为：

Data：$X = x_1, ..., x_n$

计算cost function的参数：perplexity Perp

优化参数：设置迭代次数T，学习速率η, 动量$\alpha(t)$
目标结果是低维数据表示$Y^t = y_1, ..., y_n$
开始优化
计算在给定Perp下的条件概率$p_{j|i}$(参见上面公式)
令$p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
用$N(0, 10^{-4}I)$随机初始化Y
迭代，从t = 1 到T，做如下操作:
计算低维度下的q_{ij}
计算梯度
更新每个数据点
结束

3 实验结果

本次实验的测试代码来自网上的t-SNE算法实现，该实现使用的是iris数据集，最终得到的效果图如下：

在图中，不同颜色的点表示不同的数据点，可以看到相同类别的数据被划分成不同的区域。

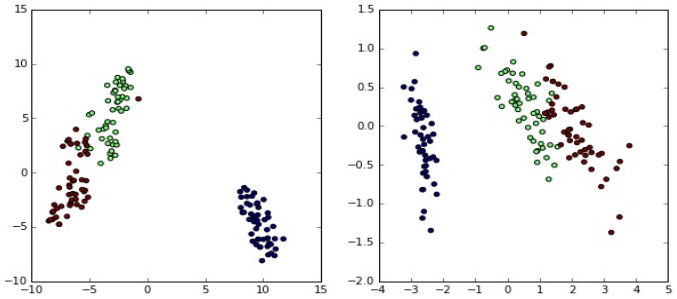

Figure 1: t-SNE算法测试效果图

4 小结与讨论

除了上述的主要流程外，t-SNE算法还有两个小技巧用来提升最终可视化的效果，第一种方式是在原有的成本函数的基础上再加一个L2惩罚项。第二种方式是把所有的p_{ij}乘一个数值，得到一个新的p_{ij}，这样会使得得到的q_{ij}的值也相应的变大。另外，t-SNE算法还存在3个缺陷：1. 无法处理其他形式的数据降维问题（比如要求降维后数据点的维度大于3维）。2.对数据的内在维度诅咒比较敏感。3. 无法得到成本函数对应的全局最优解。

参考文献

[1] J.A. Lee and M. Verleysen. Nonlinear dimensionality reduction. Springer, New York, NY, USA, 2007.