wwz

2020 数据分析

基于无监督深度学习进行谱聚类

姓名	班级	学号
王成航	统计 71	2176122248
张申铎	统计 71	2176112379
王泽昊	统计 71	2176112782

摘　要

谱聚类是无监督数据分析中的一种领先且流行的技术, 但它的两个主要限制因素是可括展性和谱嵌入的泛化问题 (即对未知数据的预测能力) . 在本文中, 我们研究了一种能克服以上缺点的深度学习谱聚类方法, 其被称为谱神经网络. 其通过训练一个孪生神经网络构造出一种表现度量, 使得优化这个表现度量能够让它替代谱映射, 把输入数据嵌入到它们邻接图拉普拉斯矩阵特征空间里, 从而将它们聚类, 功能同传统谱聚类算法. 其优点在于可以扩展到较大数据集. 并且, 由谱神经网络学习到的映射自地然对全新的未知数据完成谱嵌入与聚类的工作, 具有神经网络天生的泛化性能.

目录

1 引言

在未标记的数据中进行聚类一直以来都是机器学习领域中人们不断探索和研究的问题. 随着技术进步, 图片、文本以及其他类型的数据被大量的产生. 但通过人工对其做标记仍是一个费时费力的难题, 此类工作通常十分繁杂且需要专业知识. 而聚类技术恰巧提供了有用的工具来分析未标记的数据并揭示他们的内在结构. 在许多聚类算法中, 谱聚类则是领先且广泛流行的聚类算法. 它通过将数据嵌入拉普拉斯矩阵的特征空间, 获取数据点之间的成对相似度, 并对此应用 k-means 来聚类.

谱聚类一般具有以下几个特性:

1. 它的嵌入过程优化了损失函数, 极小化了相似数据点之间的成对距离; 此外, 还可以通过分析找到这种最佳嵌入.
2. 谱聚类的表现得优于其他流行的聚类算法, 最主要的是它具有处理非凸数据的能力.
3. 谱聚类具有可靠的成熟理论概率解释.

虽然数据点的谱嵌入可以通过对其拉普拉斯矩阵的特征分解来实现, 但对于大型数据集来说, 直接计算特征向量可能是不可能的, 即其缺少可扩展性. 此外, 将谱嵌入泛化到未知的数据, 这种问题通常被称为样本外扩展问题 (OOSE) , 是一项并不容易的任务.

谱神经网络是一种用于谱聚类的深度学习方法, 它解决了上面提到的可拓展性和样本外扩问题. 具体来说, 它借助了随机优化, 并使其具有可扩展性. 而且, 一旦经过训练, 它还能实现作为前馈网络的功能, 将每个输入数据点映射到其谱嵌入坐标. 这个映射可以轻松地应用于新的测试数据, 自然地具有了泛化性能.

与标准深度学习模型的优化不同, 谱神经网络是通过添加一个权重由 QR 分解得到的线性层来在保证正交化条件的同时, 去完成对损失函数的优化的. 在谱聚类算法中, 选择一个良好的亲和度函数对于谱聚类的结果提升是十分重要的. 相比于使用常见的基于欧几里得距离去计算高斯亲和度, 该算法在未标记样本上训练了另外一个孪生神经网络来学习改进高斯亲和度.

并且如果对数据进行自编码器的降维预处理, 可以实现性能的进一步的提升以及减少计算开销.

2 前提知识

2.1 自编码器

变分编码器是一种对高维数据进行高效特征提取与特征表示的无监督学习方法. 其能够对高维数据进行降维减轻计算的压力, 同时能够保留有用的信息, 也能达到一定的数据清洗的效果.

自编码器框架包含两大模块: 编码过程和解码过程 **图 1**. 通过编码器将输入样本映射到编码空间, 即编码过程; 然后再通过解码器将编码映射回原始空间得到重构样本, 即解码过程. 优化目标则是通过最小化重构误差来同时优化编码器和解码器, 从而学习得到针对样本输入的编码表示.

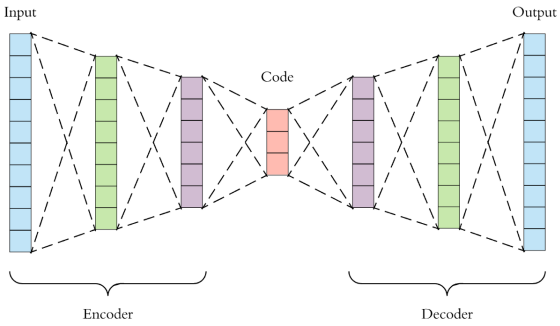


图 1: autoencoder 示意图

2.2 谱聚类算法

谱聚类是从图论中演化出来的算法, 后来在聚类中得到了广泛的应用. 它的主要思想是把所有的数据看做空间中的点, 这些点之间可以用边连接起来. 通过对所有数据点组成的图进行切图, 让切图后不同的子图间边权重和尽可能的低, 而子图内的边权重和尽可能的高, 从而达到聚类的目的.

对于一个图 G, 一般用点的集合 V 和边的集合 E 来描述, 即为 $G(V,E)$, 并定义 w_{ij} 为 v_i, v_j 之间的权重, 由于是无向图, 所以 $w_{ij} = wji$.

对于有边连接的两个点 v_i 和 v_j, $w_{ij} > 0$. 对于没有边连接的两个点 v_i 和 v_j,$w_{ij} = 0$. 对于图中的任意一个点 v_i, 它的度 d_i 定义为和它相连的所有边权重之和, 即 $d_i = \sum_{i=1}^{n} w_{ij}$.

利用每个点度的定义, 我们可以获得 $n \times n$ 的度矩阵 D, 它是一个对角矩阵, 只有对角线有值, 定义如下:

$$\begin{bmatrix} d_1 & \cdots & \cdots & \cdot \\ \vdots & d_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \cdot & \cdots & \cdots & d_n \end{bmatrix}$$

利用所有点之间的权重值, 可以获得邻接矩阵 W, 它也是一个 $n \times n$ 矩阵, 第 i 行的第 j 个值对应权重 w_{ij}. 一般来说邻接矩阵的构造思想是距离较近的两个点之间边权重较高, 距离较远的两个点之间的边权重值较低, 不过这只是定性, 我们需要定量的权重值. 一般可以通过样本点距离度量的相似矩阵 S 来获得邻接矩阵 W. 构建邻接矩阵 W 的方法有三类: ϵ-邻近法, K 邻近法和全连接法.

下面引入拉普拉斯矩阵 $L = D - W$, 其中 D 为度矩阵, W 为邻接矩阵. 对于拉普拉斯矩阵, 它具有良好的性质, 比如它是半正定对称矩阵, 所有特征值都是实数, 对于任意的向量 f, 有

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

基于以上的前提知识, 就可以正式进行切图了, 对于无向图 G 的切图, 目标应为将切成相互没有连接的个子图, 一般来说有两种切图方式： RatioCut 和 Ncut. 总结以上概念, 谱聚类的算法流程可以总结如 **算法2**.

2.3 QR 分解与 Cholesky 分解

QR 分解

对于任何一个实方阵, 存在一个正交矩阵 Q 以及一个上三角阵 R 使得

$$A = QR.$$

而对于一个一般的 $m \times n, m \geq n$ 的复矩阵 A, 存在一个酉矩阵 Q 与一个上三角阵 (底下的 $(m-n)$ 行全部是 0) 的 R 使得

$$A = QR.$$

Cholesky 分解

对于一个正定的对称矩阵 D(或者对于一个一般的矩阵 A, 考虑 A^TA), 存在一个下三角阵 L 使得

$$D = LL^T$$

我们可以通过求解 QR 分解来对一个矩阵 A 进行 Gram-Schmidt 正交化 (求得正交矩阵 Q), 并且这样的 QR 分解可以通过对 A^TA 进行 Cholesky 分解完成. 首先因为 L 是一个下三角阵, L^{-1} 也是下三角阵. 故 $(L^{-1})^T$ 是一个上三角阵. 则此时 A 的前 i 列的列空间与 $Q = A(L^{-1})^T$ 的前 i 列的列空间是相同的. 故如果我们能够证明 $Q^TQ = I$, 那么我们所求的 Q 矩阵就是这里定义的 $Q = A(L^{-1})^T$. 通过计算

$$Q^TQ = L^{-1}A^TAA(L^{-1})^T = L^{-1}LL^T(L^{-1})^T = (L^{-1}L)^T = I$$

我们发现这样对 A 的 Gram-Schidt 正交化的确可以通过 Cholesky 分解获得, 并且只需要给其右乘一个分解所得的矩阵 L 的逆的转置.

3 谱神经网络介绍

考虑一个标准的聚类问题, $\mathcal{X} = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$ 是一组从某个未知分布 $\mathcal{D}$ 中采样得出来的未标注的数据点. 在给定了类的个数 k 和一个在数据点上的距离以后, 任务是去学习一个相似的在 $\mathcal{X}$ 里抽样出来的数据点上的相似性度量, 用它来去学习一个映射将每一个数据点聚成 k 个类, 使得比较相似的点落在同样的一个聚类中. 并且更进一步地学习到一个映射, 使得其能够对新的未知的数据点进行分类, 即一个具有泛化性能的谱聚类算法. 这样的分类结果是仅仅依赖于固定的分类映射的, 在拿到新的数据点后它既不需要去计算新的点之间的相似度, 抑或是重新对加入新的点以后的数据进行重新聚类, 它可以直接输出最后的聚类结果.

对于谱聚类算法, 一般需要一个谱映射 $F_\theta : \mathbb{R}^d \to \mathbb{R}^k$ 来作为对相似度的度量, 即将数据嵌入到一个描述相似度的空间里去的映射. 以及一个在嵌入上的聚类函数 $c : \mathbb{R}^k \to \{1, \ldots, k\}$, 来完成最后的聚类工作. 谱神经网络通过构造表现度量 (由一个孪生神经网络构造得到) 使得一个神经网络能够

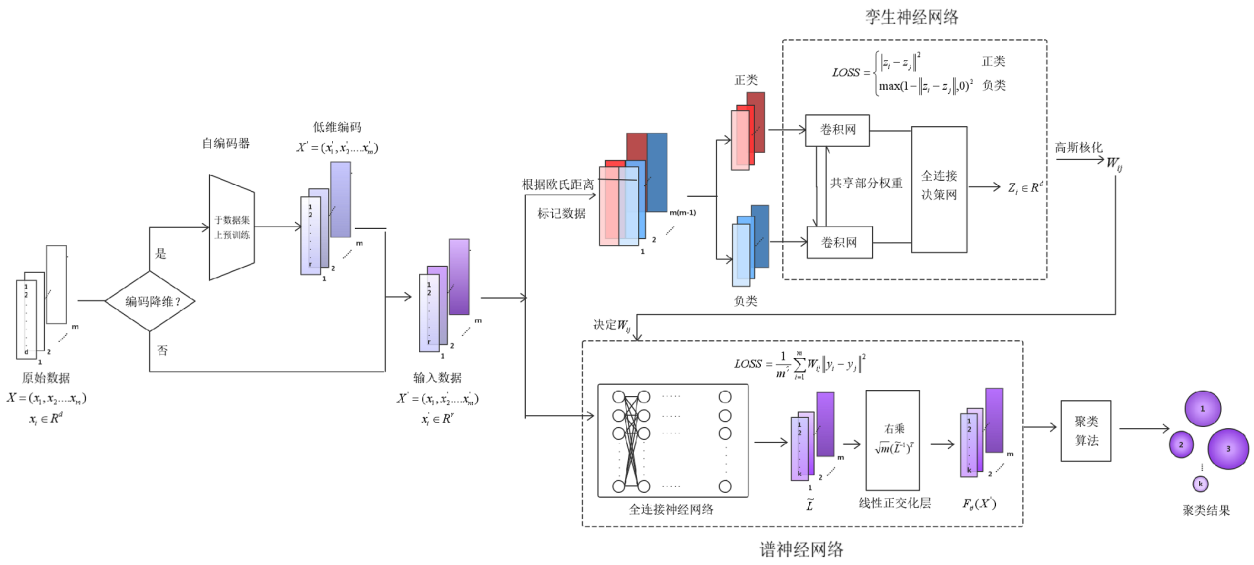


图 2: 完整的网络结构图. 其训练方式见**算法1.**

替代谱映射来对数据做嵌入 F_θ, θ 是网络的权重. 同时结合数据清洗与 k-means 算法来完成整个聚类过程, 获得了优良的泛化性能.

整个网络的训练分成三个部分,

1. 使用无监督学习训练孪生神经网络作为亲和度度量.
2. 使用孪生神经网络给出的亲和度度量来, 训练谱神经网络, 作为谱映射, 完成数据的嵌入.
3. 采用 k-means 在谱神经网络给出的数据的嵌入上做聚类.

3.1 采用自编码器做数据预处理

在谱聚类的研究中, 数据预处理是一种很常见的提升效果以及噪声稳定性的手段. 该网络在这里采取了无监督的自编码器作为数据预处理方式. 自编码器会将 $\mathcal{X}$ 上的数据映射到编码空间中, 其具有低维低噪声但是又能保留大部分与相似度有关的信息. 然后再用编码空间作为谱神经网络的输入. 该网络使用的自编码器是公开的针对数据集 MNIST 与 Reuters 的去噪自编码器. 并且通过实验对比, 我们发现在编码空间上进行聚类可以提升算法的性能, 可以极大的提高孪生神经网络的训练速度, 并且在更强大机器上完成的实验说明算法的表现也获得了巨大的提升 [1]. 该网络采用了一个从 MNIST 与 Reuters 数据集上预训练的自编码器 [2].

3.2 孪生神经网络

亲和度的度量选择是谱聚类算法成功的重要因素之一. 在常见的谱聚类算法里, 人们经常会采用在数据近邻做高斯核函数变换, 在非近邻直接令其为 0 来作为亲和度的度量, 同时将其对称化得到

高斯亲和度

$$W_{i,j} = \begin{cases} \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right), & x_j \text{ 在 } x_i \text{ 的近邻中.} \\ 0, & \text{其他,} \end{cases} \tag{1}$$

$$W_{i,j} \leftarrow \frac{(W_{i,j} + Wj,i)}{2}. \tag{2}$$

但是这样的亲和度度量过于简单, 构造与寻找其他的亲和度度往往可以提升谱聚类算法的效果.

孪生神经网络就是这样一种可以被训练去替代亲和度度量的神经网络 [3]. 简单来说, 孪生神经网络就是共享了部分权值的连体神经网络. **图 3**

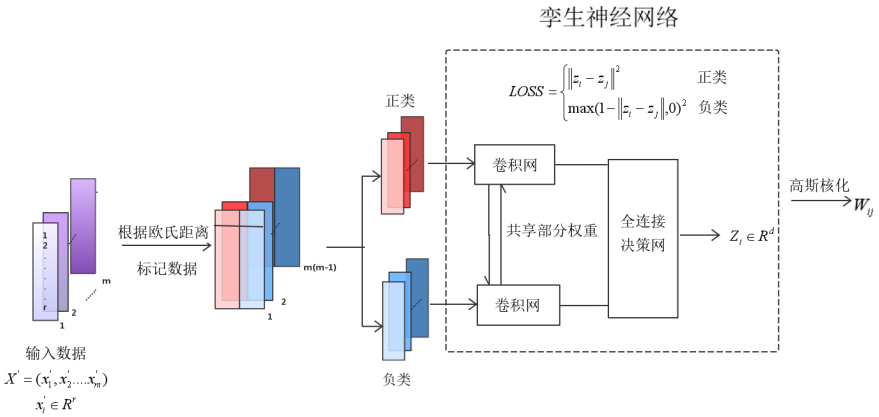


图 3: 孪生神经网络示意图. 全链接决策网是 r 个 ReLu 神经元组成的全链接神经网络.

孪生神经网络往往是在有监督的条件下训练的. 在给定了相似的样本与不相似的样本以后, 通过构造表现度量可以完成这样一个判定相似度的任务. 但是谱聚类算法研究的是无监督的情况, 数据没有标记. 在这种情况下, 孪生神经网络也有办法直接从欧几里得距离或者是图距离里学习亲和度函数. 简单来说就是通过距离函数对样本实行标记, 距离近的标记为正类, 距离远的标记为负类, 从而使其转化成一个有监督问题. 使用对比损失函数作为其表现度量

$$L_{SiameseNet}(\theta_{SiameseNet}; x_i, x_j) = \begin{cases} ||z_i - z_j||^2, & (x_i, x_j) \text{ 是正类} \\ \max(1 - ||z_i - z_j||, 0)^2, & (x_i, x_j) \text{ 是负类,} \end{cases} \tag{3}$$

其中 z_i 是我们的数据点经过孪生神经网络变换得到的在某个空间上的嵌入, 即孪生神经网络的输出. 可以看到这样的表现度量的设置促使神经网络逐渐把正类嵌入到彼此附近, 同时把反类嵌入到相对较远位置.

先训练孪生神经网络, 当其训练完成后, 用其去在每个 minibatch 上去定义亲和度矩阵 W. 具体而言, 将 1 里的 $||x_i - x_j||$ 替换为样本在孪生神经网络的输出 $||z_i - z_j||$, 以此来作谱神经网络的损失函数里的亲和度 W.

3.3 谱神经网络的表现度量与优化算法

对于一个对称的亲和度函数 $w : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$, 数据点 x, x' 之间的亲和度可以由 $w(x, x')$ 进行表达. 如果这样的一个亲和度函数给定, 在这个函数下相似度比较大的点在谱映射了以后需要被嵌入到彼此的附近. 所以定义损失函数

$$\mathcal{L}(\theta) = \mathbb{E}[w(x, x')||y - y'||^2] \tag{4}$$

其中 $y, y' \in \mathbb{R}^k$, 期望 $\mathbb{E}$ 是对每一个从数据集 $\mathcal{D}$ 中抽样出来的数据对进行的均值运算, θ 是网络的参数. 为了避免退化的平凡解, 即将所有样本点都映射到同一点上, 要求该网络的输出 y 对于分布 $\mathcal{D}$ 的期望 (实际上采用其的经验分布替代, 即期望由均值代替) 彼此正交,

$$\mathbb{E}[yy^T] = I_{k\times k}. \tag{5}$$

优化算法采取随机优化的方式, 随机抽取 m 个样本 $x_1, \ldots, x_m \in \mathcal{X}$ 作为一个 minibatch, 将他们作为行向量组成 $m \times d$ 的矩阵 X. 同时定义谱映射的输出 $y_i = F_\theta(x_i)$, 以其作为行向量组成 $m \times k$ 矩阵 Y. 记 W 是 $m \times m$ 的相似度矩阵, 其有 $W_{i.j} = w(x_i, x_j)$. 那么此时损失函数与正交性要求就变成了

$$L_{SpectralNet}(\theta) = \frac{1}{m^2}\sum_{i,j=1}^{m} W_{i,j}||y_i - y_j||^2, \tag{6}$$

$$\frac{1}{m^2}Y^TY = I_{k\times k}. \tag{7}$$

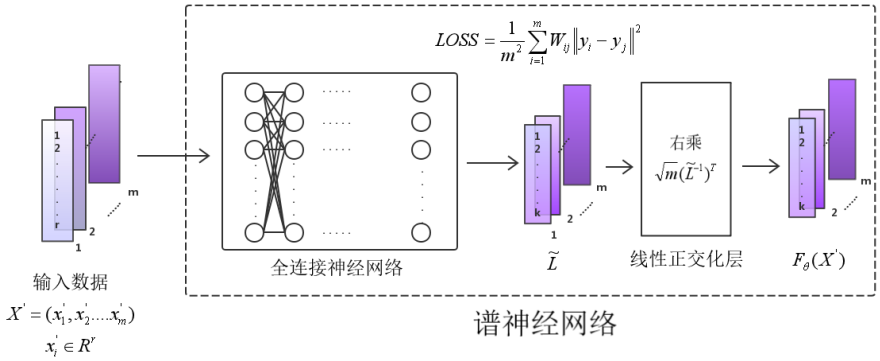


图 4: 谱神经网络结构图.

该谱聚类算法的谱映射是由一个神经网络, 谱神经网络, 完成的. 这个神经网络由两部分组成, 一部分是普通的全连接神经网络, 另一部分是最后一层正交化层. 这一层采用了有 k 个输入 k 个输出的线性层, 并且网络权重被设置成了能够正交化输出的一种组合. 假设 $\tilde{Y}$ 是一个 $m \times n$ 的矩阵, 作为最后一层正交化层的输入, 此时采用 QR 分解来生成能够将其列正交化的线性映射. 考虑 $\tilde{Y}$ 的 Cholesky 分解 $\tilde{L}$, 则去正交化 $\tilde{Y}$ 等价于在最后一层线性层右乘 $\tilde{Y}$ 以 $\sqrt{m}(\tilde{L}^{-1})^T$.

对于谱神经网络的训练, 每一次训练都采用不同的 minibatch, 其均匀地从 $\mathcal{X}$ 中抽出. 该网络的训练过层也分为两个部分, 首先是根据数据点用 QR 分解生成最后的正交化层. 接着使用通常的反向传播算法来完成梯度下降. 最后采用 k-means 聚类算法作为聚类映射.

3.4 与谱聚类的联系

注意到 **损失函数 6** 可以被重写为

$$L_{SpectralNet}(\theta) = \frac{2}{m^2}\text{trace}(Y^T(D_W)Y),$$

其中 D 是一个对角阵, 其对角线元素 $D_{i,i} = \sum_j W_{i,j}$. 此时, $D - W$ 是一个对称的半正定矩阵, 它成为了每个 minibatch 里数据的图拉普拉斯矩阵 (未归一化的). 去在正交的约束下优化这个损失函数, 在 Y 的列空间是 D_W 的最小的 k 个特征值的特征向量所张成的特征子空间的时候, 目标损失函数达到最小. 从这个角度去看, 谱神经网络通过构造的表现度量实际上近似了谱聚类的过程.

其与传统谱聚类算法最主要的却别在于该网络在全数据上的正交性只是近似的正交性 (因为只有在 minibatch 上面的严格正交性), 并且优化算法是采用了随机优化的办法. 相对于传统的谱聚类算法, 谱神经网络虽然损失了严格的正交性, 但是其却获得了更好的泛化性能以及在数据量庞大的情况下的时候的适用性. 传统的谱聚类算法在数据集庞大的情况下去计算真实准确的特征向量需要耗费极大的计算资源, 耗费到以至于谱聚类无法使用. 而对于正交性的宽松以及采用随机优化可以在大数据集上使用这个方法. 并且更进一步对于未出现的数据点, 该网络所学习的的映射也可以自然的直接对新的数据点进行判断, 而不需要重新计算聚类.

最后一步, 与传统的谱聚类算法在得到数据的嵌入 $y_1, \ldots, y_n$ 了以后一样地采取了 k-means 作为聚类的方法. 但是 k-means 也可以被替换为其他的聚类算法.

3.5 谱神经网络训练流程

谱神经网络的具体的端到端的训练流程如 **算法1** 所示.

4 训练结果

4.1 训练准备与自编码器带来的训练性能飞跃

我们在一台配备了 AMD Ryzen R9 3700X, 128G 内存与一张 1080Ti 显卡的服务器上进行了实验. 第一次我们选择了一台 32G 内存的服务器进行实验, 但是内存太小导致 Keras 报错, 于是我们更换了 128G 的服务器, 发现这个模型大概耗费掉了约 100G 的内存. 总共的训练在服务器上运行了大概三个小时左右.

并且在我们实验的过程中, 我们发现我们觉得已经算足够强大的计算机甚至不足以在不进行自编码器降维的条件下, 对原始数据完成模型的训练. 我们这么强大的一台服务器运行了两个小时甚至没有完成第一个孪生神经网络的训练. 我们已经使用了相对高端的机器却依然在高维数据的训练上遇到了困难, 由此可见该自编码器的应用对可拓展性的提升是从不可能到可能的巨大飞跃.

最后我们在数据集 MNIST 上展开训练, 将预训练的自编码器提取出来的特征数据作为两个网络的训练数据集输入

并且我们采取两种常见的性能度量来对谱网络进行评价, 无监督聚类准确率与归一化互信息. 其中无监督聚类准确率是指, 对于数据 x_i, 如果其真实类属与预测标签分别为 l_i 和 c_i. 记向量 $l = (l_1, \ldots, l_n)$ 与 $c = (c_1, \ldots, c_n)$.

则无监督准确率 ACC 为

$$ACC(l, c) = \frac{1}{n} \max_{\pi \in \Pi} \sum_{i=1}^{n} \mathbf{1}\{l_i = \pi(c_i)\},$$

其中 Π 是 $\{1, \ldots, k\}$ 的排列.

归一化互信息 NMI 为

$$NMI(l, c) = \frac{I(l; c)}{\max\{H(l); H(c)\}},$$

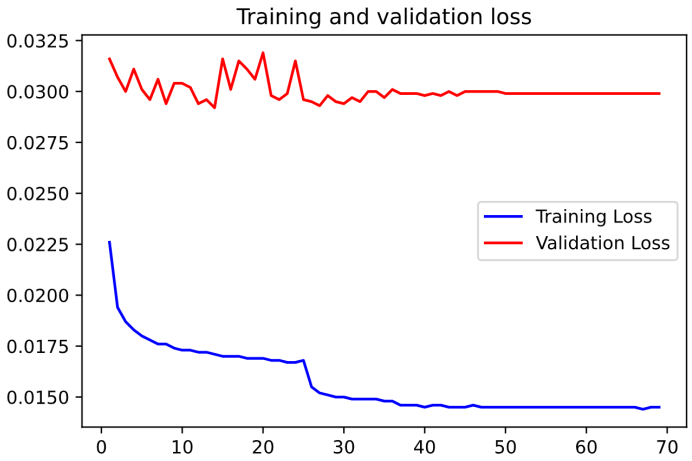
其中 $I(l; c)$ 表示 l 与 c 的互信息, $h(\cdot)$ 是熵函数. 两个指标都是 [0,1] 之间的数, 并且越大说明聚类的结果越好.

4.2 孪生神经网络的训练

将原数据的 1/10 作为验证集, 设置参数如下

Epochs	Learning rate	Learning rate decay	Optimizer
400	$1e^{-3}$	0.1	RMSprop

训练过程中在训练集与验证集上的损失函数如下



我们采用了 Keras 里的 early stop 判定, 其在训练看不到提升的时候就认为模型已经收敛, 可以自动停止训练防止模型过拟合. 我们可以看到其并没有训练满我们预定的 400 个 Epoch. 总共训练了 69 个 Epoch 就自动停止了. 此时我们人眼也可以看出模型已经收敛.

接下来便使用此网络的输出结果作为谱神经网络的亲和度函数.

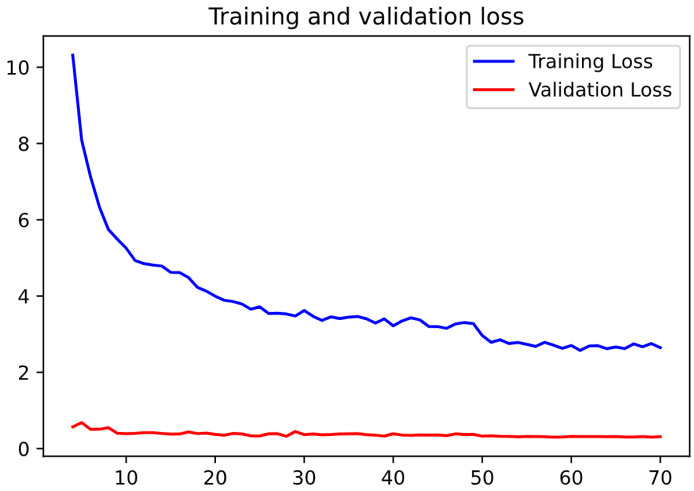


图 5: 谱神经网络训练过程的损失函数, 只画出了前 70 个 Epoch, 实际上在 Early stop 前进行了 221 个 Epoch, 70 个以后 Validation Loss 跟 Training Loss 已经趋于稳定, 人眼无法分别其波动.

4.3 谱神经网络的训练

此网络的主要超参数设置与孪生神经网络相同, 训练过程如 **图 5** 所示. 我们依旧采用了 early stop 功能来防止过拟合. 在 70 个 Epoch 以后网络从肉眼来看就已经收敛, Validation Loss 与 Training Loss 都趋于定值. early stop 功能在训练到 221 个 Epoch 的时候自动停止了训练. 为了表现出损失函数的下降过程, 我们选择只展示前 70 个 Epoch 的训练过程里损失函数的变化. 可以看到, 在验证集上的损失函数值是十分低的, 模型收敛得很好.

4.4 谱神经网络训练结果与最终聚类结果

谱神经网络的训练结果与测试结果如 **图 6** 所示,

```
Epoch: 215, loss=2.500463, val_loss=0.295505
Epoch: 216, loss=2.586550, val_loss=0.300419
Epoch: 217, loss=2.540537, val_loss=0.295834
Epoch: 218, loss=2.496027, val_loss=0.297053
Epoch: 219, loss=2.522836, val_loss=0.297530
Epoch: 220, loss=2.569293, val_loss=0.293700
Epoch: 221, loss=2.539236, val_loss=0.303620
STOPPING EARLY
finished training
confusion matrix:
[[   2    8 6840    1    3   29    8    5    7    0]
 [  12    5    1 7666   22    5    8  129    7   22]
 [  30   21   29    5    4    4   23 6865    0    9]
 [  30 6933    2    7   16    4   58   54   37    0]
 [   9    1    3    9  190   28    5    7    2 6570]
 [   4  256   17    2   32   51   50    7 5893    1]
 [   0    1   26    6    3 6770   13    4   52    1]
 [7091    4    3   31  107    0    5   40    2   10]
 [  10  129   10   16   34   23 6507   28   57   11]
 [  50  118   31    8 6665    7   26    6   14   33]]
spectralNet accuracy: 0.969
/usr/local/miniconda3/envs/dl/lib/python3.5/site-packages/sklearn/metrics/cluster/supervised.py:844: FutureWarning: The behavi
or of NMI will change in version 0.22. To match the behavior of 'v_measure_score', NMI will use average_method='arithmetic' by
 default.
  FutureWarning)
NMI: 0.92
(dl) root@9cdf19cff90b:/data/A/src/applications#
```

图 6: 谱神经网络训练的训练结果. 其中包含模型的训练轮数, 最终的损失函数值, 分类的混淆矩阵以及准确率和归一化互信息等指标.

得到的混淆矩阵如下

2	8	6840	1	3	29	8	5	7	0
12	5	1	7666	22	5	8	129	7	22
30	21	29	5	4	4	23	6865	0	9
30	6933	2	7	16	4	58	54	37	0
9	1	3	9	190	28	5	7	2	6570
4	256	17	2	32	51	50	7	5893	1
0	1	26	6	3	6770	13	4	52	1
7091	4	3	31	107	0	5	40	2	10
10	129	10	16	34	23	6507	28	57	11
50	118	31	8	6665	7	26	6	14	33

最后训练了 221 个 Epoch, 达成了在测试集上的 96.9% 的准确率. 得到最终结果与真实的分类结果的归一化互信息 (NMI) 为 0.92, 这与准确率都说明此算法的聚类结果还算不错.

4.5 孪生神经网络

我们之前提到孪生神经网络只不过是一种亲和度函数的替代, 我们还测试了另外一种很常用的无监督模型, KNN 模型, 作为我们的亲和度函数. 我们希望对比这两种无监督模型哪个作为亲和度函数在 MNIST 数据集上的表现更好.

我们使用相同的超参数在 MNIST 数据集上进行训练, 训练过程中训练集与验证集上的损失函数如 **图 7** 所示.

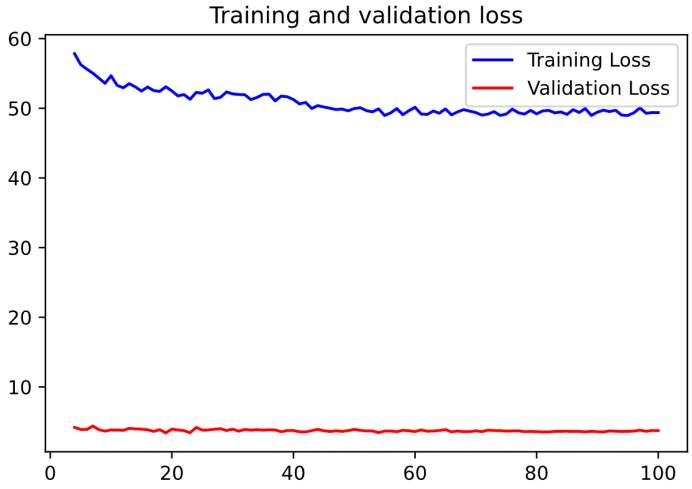


图 7: 训练过程中的损失函数.

最终的混淆矩阵如 1 所示.

12	11	71	148	4	6522	1	4	112	18
20	14	3	5	5	0	4230	3590	10	0
256	5886	68	34	96	52	171	188	238	1
96	36	215	9	57	1	67	18	6641	1
2147	2	15	38	4410	3	78	121	1	9
18	2	3313	82	120	18	33	17	2706	4
2	0	29	4397	23	35	14	24	37	2315
6624	15	5	0	392	4	113	136	4	0
170	11	3835	18	93	21	113	94	2465	5
3326	6	13	3	3392	22	33	24	136	3

表 1: KNN 结果的混淆矩阵

此时, 得到最终结果与真实的分类结果的归一化互信息 (NMI) 为 0.67, 准确率为 79.4%. 这样的结果说明 KNN 也可以作为谱神经网络的亲和度度量. 但是在 MNIST 数据集上, 其表现远远不如孪生神经网络来的好, 孪生神经网络更适合作为谱神经网络的亲和度度量. 不过究竟有没有一种数据集使得 KNN 在上面能够胜过孪生神经网络呢? 有没有比孪生神经网络更适合做谱神经网络的亲和度度量的非监督模型呢? 我们不敢妄下定论.

参考文献

[1] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017.

[2] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: A generative approach to clustering. *CoRR*, abs/1611.05148, 2016.

[3] Uri Shaham and Roy R Lederman. Learning by coincidence: Siamese networks and common variable learning. *Pattern Recognition*, 74:52–63, 2018.

[4] Boaz Nadler, Ronen Basri, Yuval Kluger, Uri Shaham, Kelly Stanton, and Henry Li. Spectralnet: Spectral clustering using deepneural networks. *arXiv*, 1801.01587v6, April 2018.

[5] 谱聚类（spectral clustering）算法介绍. `https://blog.csdn.net/qq_24519677/article/details/82291867`.

[6] Siamese network 孪生神经网络. `https://www.jianshu.com/p/92d7f6eaacf5`.

[7] 归一化互信息 (nmi) 评价指标. `https://blog.csdn.net/hang916/article/details/88783931`.

[8] Qr decomposition. `https://en.wikipedia.org/wiki/QR_decomposition`.

[9] Chelosky decomposition. `https://en.wikipedia.org/wiki/Cholesky_decomposition`.

A 算法流程

A.1 谱神经网络聚类算法流程

Algorithm 1 谱神经网络聚类算法

输入: $\mathcal{X} \in \mathbb{R}^d$, 需要聚的类的个数 k, 每一个 batch 的大小 m.

输出: 谱嵌入 $y_1, \ldots, y_n, y_i \in \mathbb{R}^k$, 聚类结果 $c_1, \ldots, c_n, c_i \in \{1, \ldots, k\}$.

针对数据使用预训练的自编码器做编码预处理;
根据欧式距离构造用于训练孪生神经网络的正类数据与反类数据;
训练一个孪生神经网络;
while $\mathcal{L}_{SpectralNet}$ 不收敛 **do**
    正交化步:
    从 $\mathcal{X}$ 中随机抽样一个含有 m 个样本的 minibatch 记作 X;
    前向传播 X, 计算出正交层的输入 $\tilde{Y}$;
    计算 Cholesky 分解, $LL^T = \tilde{Y}^T\tilde{Y}$;
    将正交化层的权重设置为 $\sqrt{m}(L^{-1})^T$;
    梯度下降步:
    从 $\mathcal{X}$ 中随机抽样一个含有 m 个样本的 minibatch 记作 X;
    用孪生神经网络计算出 $m \times m$ 的亲和度矩阵 W;
    前向传播样本并计算损失函数6;
    对 F_θ 的非正交化层的权重进行梯度下降
正向传播所有样本 $x_1, \ldots, x_n$ 得到谱嵌入 $y_1, \ldots, y_n$;
在谱嵌入 $y_1, \ldots, y_n$ 上进行 k-means 聚类并且得到聚类中心

A.2 谱聚类算法流程

Algorithm 2 谱聚类的算法流程

输入: 样本集 $D = (x_1, x_2, \ldots x_n)$, 相似矩阵生成方式, 降维维度 k_1, 聚类方法, 聚类后维度 k_2.

输出: 簇划分 $C(c_1, c_2, \ldots c_{k_2})$.

根据输入的相似性矩阵的生成方式构建样本的相似矩阵 S;
根据相似矩阵 S 构建邻接矩阵 W, 构建度矩阵 D;
计算拉普拉斯矩阵 L;
进行切图;
用输入的聚类方法聚类, 聚类维数为 k_2;
到簇划分 $C(c_1, c_2, \ldots c_{k_2})$.