

A survey of graph convolution

- References
- 理论篇
 - 为什么要研究GCN
 - 回顾CNN
 - 怎么把卷积推广到非Euclidean数据上
 - GCN技术概述
 - 频域图卷积
 - graph上的傅立叶变换
 - 第一代GCN
 - 第二代GCN
 - 第三代GCN
 - 类比
 - 空域图卷积 (TODO)
- 应用篇
 - GCN在人脸识别中的应用
 - GCN在物体分割中的应用
 - GCN在点云分类中的应用
- 思考

References

论文按时间顺序排列

描述	年份	题目	链接
graph上的傅立叶变换	2012	The Emerging Field of Signal Processing on Graphs	https://arxiv.org/pdf/1211.0053.pdf
第一代GCN	2014, NIPS	Spectral Networks and Deep Locally Connected Networks on Graphs	https://arxiv.org/pdf/1312.6203.pdf
第二代GCN	2016, NIPS	Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering	http://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.pdf
第三代GCN	2017, ICLR	SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS	https://openreview.net/pdf?id=SJU4ayYgl
综述	2017	Geometric deep learning: going beyond Euclidean data	https://arxiv.org/pdf/1611.08097.pdf

理论篇

为什么要研究GCN

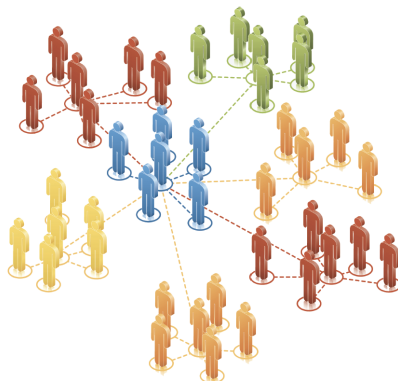
图片，语音，文字等数据，被称为 **Euclidean 数据** 或者网格数据。这种结构的数据可以有效地被CNN处理。

图或者三维几何等数据，被称为 **非Euclidean 数据**，由于 **感受野** 以及 **层级网络结构** 无法被很好的设计，CNN难以在这类数据上应用。

GCN就是由于这个原因被提了出来，现在GCN已经被广泛用于社交网络，mesh，点云等数据的处理上。



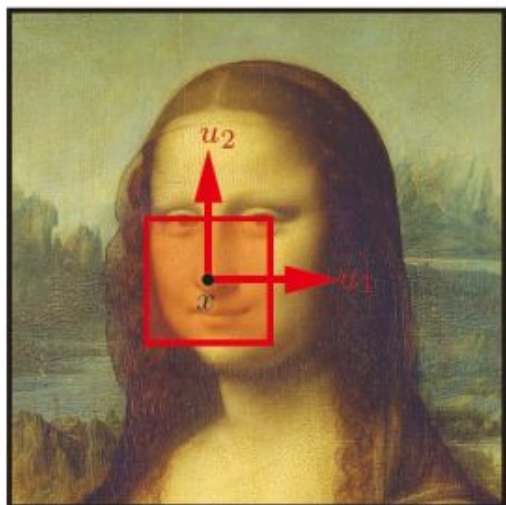
Manifolds



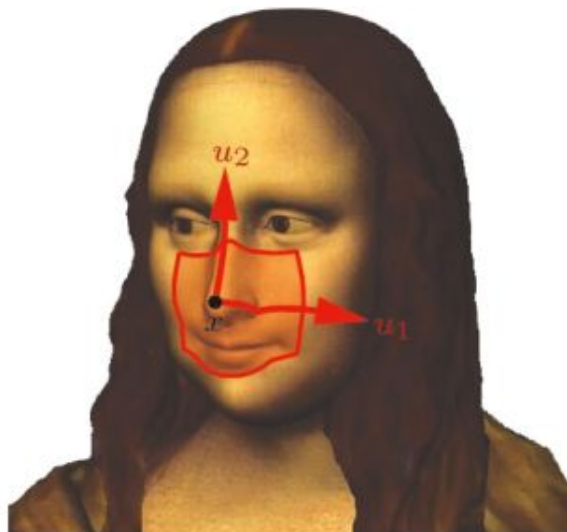
Graphs

CNN无法处理Non Euclidean Structure的数据，学术上的表达是传统的离散卷积在Non Euclidean Structure的数据上无法保持平移不变性。通俗理解就是在拓扑图中每个顶点的相邻顶点数目都可能不同，那么当然无法用一个同样尺寸的卷积核来进行卷积运算。

由于CNN无法处理Non Euclidean Structure的数据，又希望在这样的数据结构（拓扑图）上有效地提取空间特征来进行机器学习，所以GCN成为了研究的重点。



Image



Manifold
知乎 @superbrother

回顾CNN

CNN为什么比简单的堆叠 FC 有效：

1. 卷积核参数共享。利用了数据的自相似性和平移不变形。
2. 每个卷积核都作用于局部空间，可以获取更精细的特征。
3. 多尺度。使用pooling后，随着层数增加，尺度递增，特征也更趋于语义（high level）特征。
4. 每个卷积核是 $O(1)$ 复杂度，跟输入大小无关。可以提高泛化能力。

怎么把卷积推广到非Euclidean数据上

1. 怎么捕捉局部特征。（怎么定义卷积）
2. 怎么运用多尺度结构。（怎么定义pooling）
3. 怎么加速运算。（怎么把运算复杂度降低）
4. 怎么减少参数量。（怎么参数共享）

GCN技术概述

现在常用的GCN可以被分成两类，频域图卷积（谱图卷积）和空域图卷积。频域图卷积是先把信号转换到傅立叶空间，然后做卷积运算后再转换到原始空间。空域图卷积直接在图上做卷积运算。

频域图卷积

频域图卷积背后有一套理论，主要讲述怎么把傅立叶变换推广到 graph 上，下面先对该理论进行概述。

graph上的傅立叶变换

深入理解傅立叶变换可以参考 <https://tracholar.github.io/math/2017/03/12/fourier-transform.html>

傅立叶变换： $F(\omega) = \int f(t)e^{-i\omega t} dt$

傅立叶变换就是时域信号 $f(t)$ 与基函数 $e^{-i\omega t}$ 在时间上的积分。为什么选择 $e^{-i\omega t}$ 作为基函数，第一个原因： $e^{-i\omega t}$ 是正交函数系，第二个原因： $e^{-i\omega t}$ 是拉普拉斯算子(Δ)的特征函数。

拉普拉斯算子：

$$\Delta = \frac{\partial^2}{\partial x^2}$$

拉普拉斯算子特征方程：

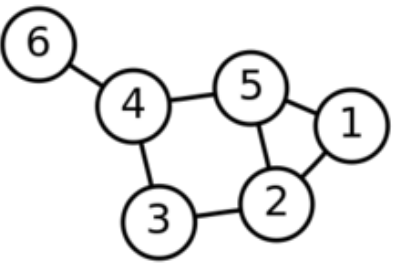
$$\Delta g = \lambda g$$

$g = e^{-i\omega t}$ 是该方程的一个解，其中特征值 $\lambda = -\omega^2$ 。

可以得到结论：**傅立叶变换就是时域信号与拉普拉斯算子特征函数的乘积在时间上的积分。**

推广：graph的傅立叶变换等于graph上的信号与graph拉普拉斯矩阵特征向量的求和？

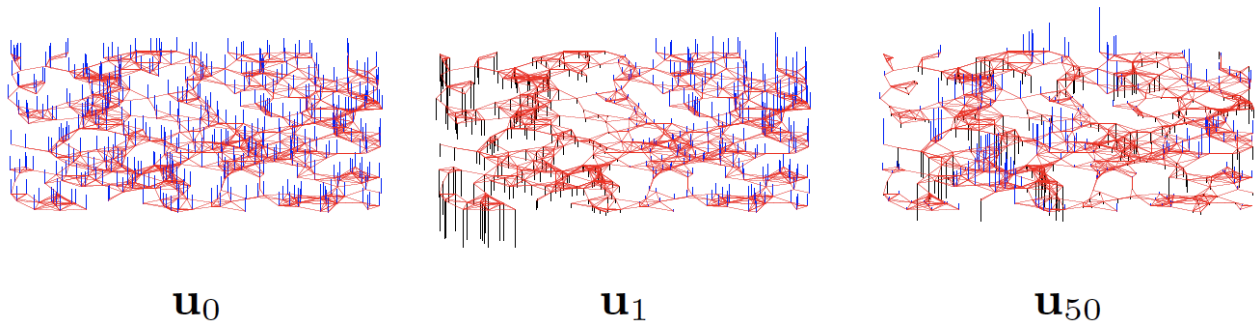
图的拉普拉斯矩阵： $L = D - A$ （[为什么这么定义](#)）。其中D是度矩阵，A是邻接矩阵。

图	度矩阵	邻接矩阵	拉普拉斯矩阵
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & & & & & \\ & -1 & & & & \\ & 0 & & & & \\ & 0 & & & & \\ & -1 & & & & \\ & 0 & & & & \end{pmatrix}$

拉普拉斯矩阵是对称矩阵，可以进行特征分解（谱分解）

$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^T$$

假定 $\lambda_1 \dots \lambda_N$ 从小到大排序，其对应的特征向量为 $u_1 \dots u_N$ ，特征值越小，对应的特征向量越平稳，这和傅立叶变换中频率的定义是类似的。下图是对 random sensor network 做特征值分解后的特征向量分布展示，可以看到特征值越小，对应的特征向量越平滑。



由上得，对于一个N点graph，定义在graph上的信号为 f ，那么对 f 的傅立叶变换为：

$$\hat{f}(\lambda_l) = \sum_{i=0}^N f(i) \cdot u_l(i)$$

其中 λ_l 是拉普拉斯矩阵第 l 个特征值， u_l 是与之对应的特征向量。把每一个分量都展开，可以得到：

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix}$$

写成矩阵的形式，有：

$$\hat{f} = U^T f$$

类似傅立叶逆变换，我们可以得到graph信号的逆变换形式为：

$$f = U \hat{f}$$

第一代GCN

我们知道，~~时域的卷积等价于傅立叶变换后的乘积再做一次逆变换~~。有了graph上的傅立叶变换定义后，对于graph上的信号 f 和卷积核 h ，可以定义graph上的卷积如下：

$$f \star h = U(U^T h) \odot (U^T f)$$

记 $\hat{h}(\lambda_l) = \sum_{i=0}^N h(i) \cdot u_l(i)$ ，上式可以写作：

$$f \star h = U \begin{bmatrix} \hat{h}(\lambda_1) & & & \\ & \hat{h}(\lambda_2) & & \\ & & \dots & \\ & & & \hat{h}(\lambda_N) \end{bmatrix} (U^T f)$$

到此，graph上的卷积就定义好了，严格按照了~~时域的卷积等价于傅立叶变换后的乘积再做一次逆变换~~理论进行推导得到。

在深度学习中，卷积核是可学习参数，如果直接另 $\hat{h}(\lambda_l) = \theta_l$ ，就可以得到**第一代GCN**：

$$f \star h = U \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \dots & \\ & & & \theta_N \end{bmatrix} (U^T f)$$

虽然利用上式已经可以构造深度网络进行图卷积运算了，但该版本有不少缺点：

1. 没有local信息。每次卷积都是所有顶点都参与运算。
2. 运算量大。每次卷积都要进行矩阵相乘，计算复杂度为 $O(N^3)$ 。
3. 参数量大。每个卷积核参数数量为 $O(N)$ 。

第二代GCN

为了改进第一代GCN，或者说把CNN里的特性引入GCN，第二代GCN被提了出来。

另 $\hat{h}(\lambda_l) = \sum_{j=0}^K \alpha_j \lambda_l^j$ ，可得卷积表达式为：

$$f \star h = U \begin{bmatrix} \sum_{j=0}^K \alpha_j \lambda_1^j & & & \\ & \sum_{j=0}^K \alpha_j \lambda_2^j & & \\ & & \dots & \\ & & & \sum_{j=0}^K \alpha_j \lambda_N^j \end{bmatrix} (U^T f)$$

而：

$$\begin{bmatrix} \sum_{j=0}^K \alpha_j \lambda_1^j & & & \\ & \sum_{j=0}^K \alpha_j \lambda_2^j & & \\ & & \dots & \\ & & & \sum_{j=0}^K \alpha_j \lambda_N^j \end{bmatrix} = \sum_{j=1}^K \alpha_j \Lambda^j$$

所以：

$$U(\sum_{j=0}^K \alpha_j \Lambda^j) U^T f = \sum_{j=0}^K \alpha_j L^j f$$

其中，K是可以选定的参数，它的值决定了感受野大小。因为当图中两个点的通路长度大于j时， $L^j = 0$ 。

上式每次卷积是矩阵和向量的乘法，复杂度为 $O(KN^2)$ ，相比第一代GCN也引入了local 的操作方式。

通过使用切比雪夫多项式对 L^j 展开，可以把复杂度降到 $O(K|E|)$ ，其中 $|E|$ 是图中边的数量。

切比雪夫多项式展开如下：

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

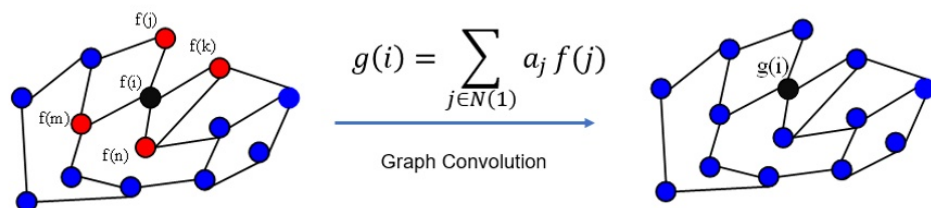
为了避免多项式迭代过程因为连乘导致数值爆炸，对拉普拉斯矩阵做如下处理：

$$\tilde{L} = 2L/\lambda_{max} - I_N$$

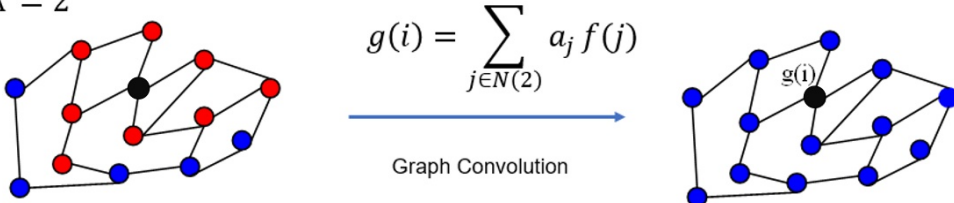
该式子作用是把 L 的谱半径约束到 [-1, 1] 之间。

操作方式：

$K = 1$



$K = 2$



总结第二代GCN优点如下:

1. 运算量相比第一代的 $O(N^3)$ 可以降低到 $O(K|E|)$ 。
2. 引入K-hop感受野, 可以捕捉局部特征。

第三代GCN

第三代相比第二代改动不大, 只是另 $K=1$, 相当于只考虑1-hop 邻点。通过堆叠层数来增加感受野。

总结:

1. 只考虑1-hop邻点, 然后通过堆叠层数增加感受野
2. 计算复杂度为 $O(|E|)$

类比

global	local
傅立叶变换	小波变换
FC	CNN
第一代GCN	第二/三代GCN

空域图卷积 (TODO)

应用篇

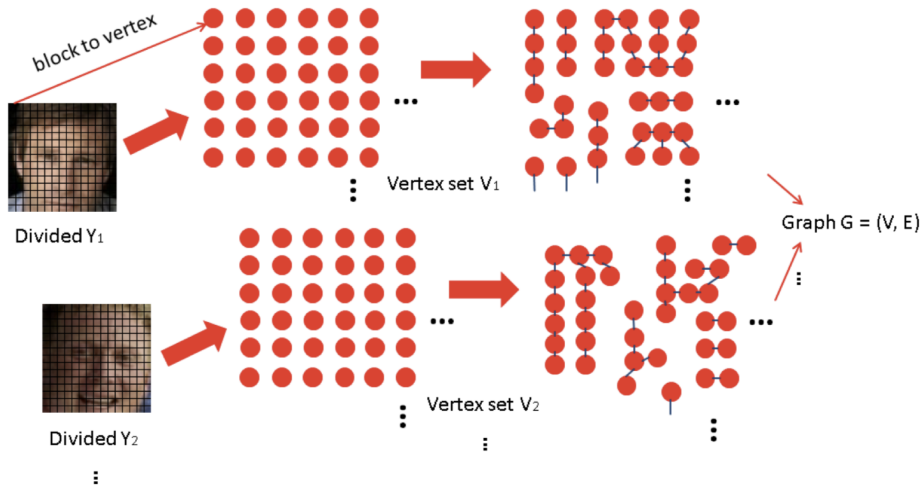
GCN在人脸识别中的应用

[Face Recognition via Deep Sparse Graph Neural Networks](#)

第一步 —— 构图

将人脸切分成block, 然后用block间的相关性代表edge:

$$w_{\alpha\beta} = \begin{cases} e^{-\frac{\|y_{\alpha}-y_{\beta}\|_2^2}{2\sigma^2}}, & \text{if } v_{\alpha} \in N_k(v_{\beta}) \text{ or } v_{\beta} \in N_k(v_{\alpha}), \\ 0, & \text{otherwise,} \end{cases}$$



第二步 —— 稀疏表示

利用字典学习对graph进行稀疏化:

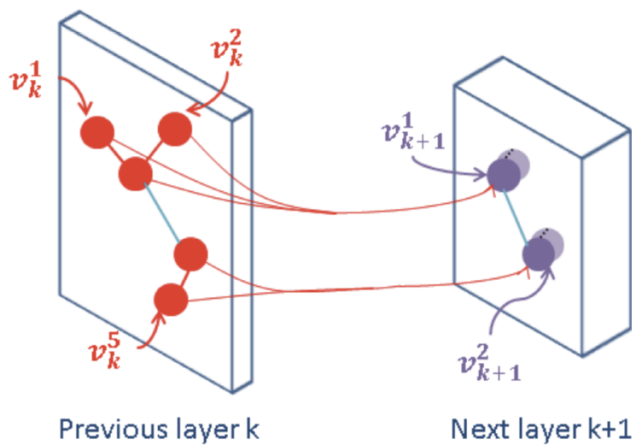
$$\min_{s, \lambda} \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\mathbf{S}\|_F^2 + \lambda \|\mathbf{S}\|_1$$

第三步 —— 网络

利用第一代GCN对构建好的graph进行训练。

$$\hat{\mathbf{y}}_{k+1,l}^q = \delta(\sum_{p=1}^P \mathbf{\Phi} \mathbf{F}_{k,p,q} \mathbf{\Phi}^T \hat{\mathbf{y}}_{k,l}^p),$$

采用了maxpooling（不合理），pooling后缺失的node补零。



结果

Table 1: Face recognition rates (complete face test) on LFW database.

Method	rates
High-dim LBP	0.9517
DeepFace	0.9735
FaceNet [10]	0.9963
Parkhi's approach [10]	0.9865
DeepID2+ [20]	0.9947
DeepID3 [15]	0.9953
DSGNN	0.9958

GCN在物体分割的应用

Beyond Grids: Learning Graph Representations for Visual Recognition

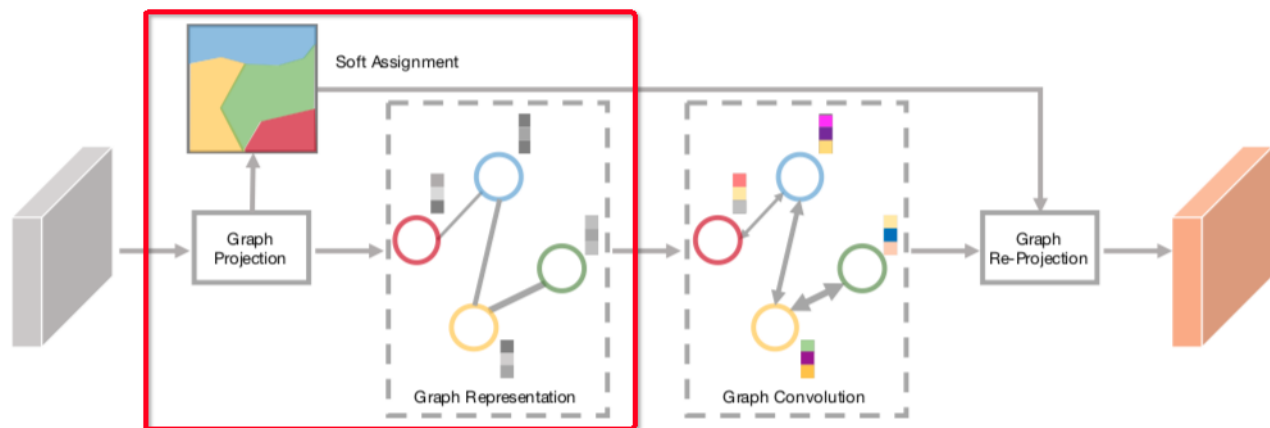
第一步 —— 构图

根据特征之间相似度进行构图:

$$q_{ij}^k = \frac{\exp \left(-\|(x_{ij} - w_k)/\sigma_k\|_2^2/2 \right)}{\sum_k \exp \left(-\|(x_{ij} - w_k)/\sigma_k\|_2^2/2 \right)},$$

$$z_k = \frac{z'_k}{\|z'_k\|_2}, \quad z'_k = \frac{1}{\sum_{ij} q_{ij}^k} \sum_{ij} q_{ij}^k (x_{ij} - w_k) / \sigma_k.$$

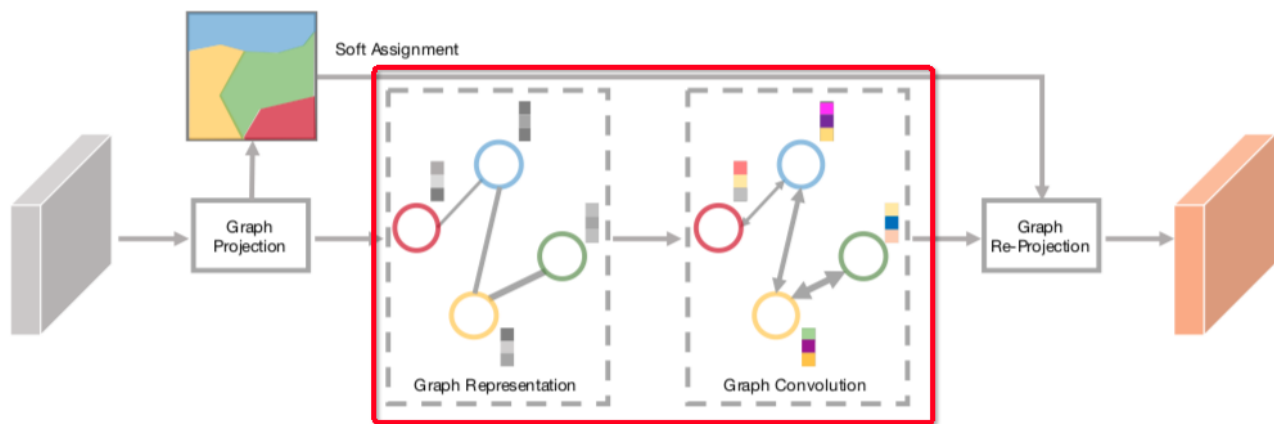
$$A = Z^T Z.$$



第一步 —— 计算

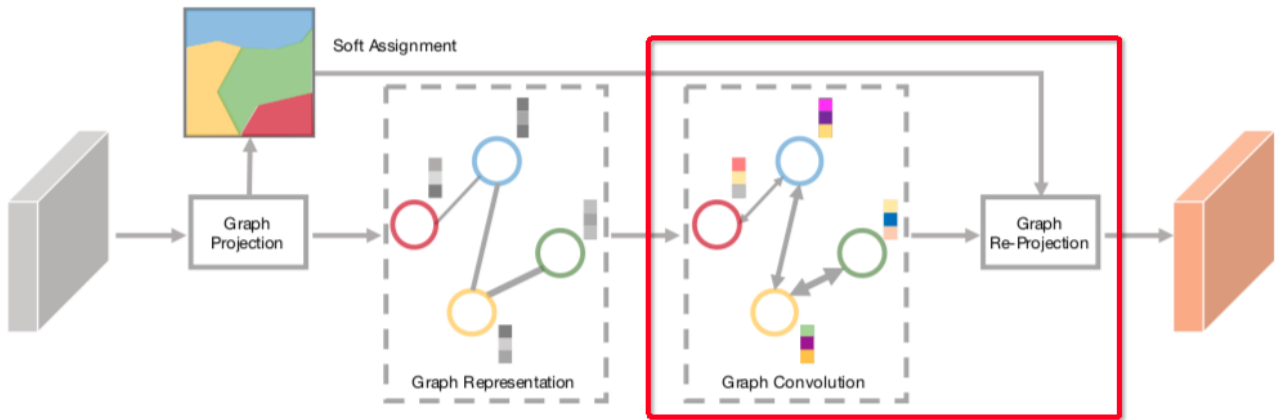
用图卷积对构好的图进行操作

$$\tilde{Z} = f(\mathcal{A}Z^T W_g),$$



第一步 —— 反映射

图中每个节点特征反映射为原始的CNN featuremap



GCN在点云分类中的应用

A GRAPH-CNN FOR 3D POINT CLOUD CLASSIFICATION

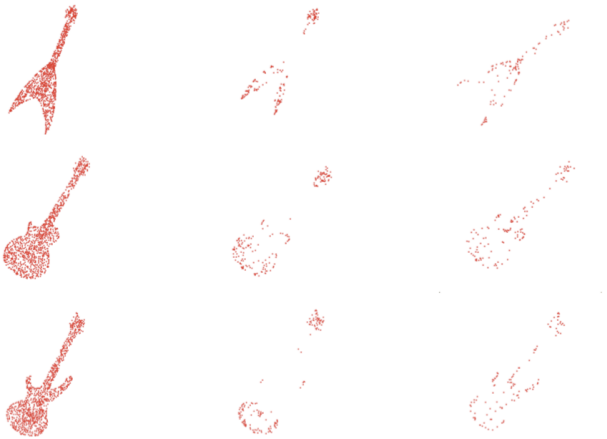
第一步 —— 构图

对于3D点云，利用高斯核计算两点之间距离作为边：

$$W_{i,j} = \begin{cases} \exp(-\|x_i - x_j\|^2 / \sigma^2) & \text{if } j \in \mathcal{A}_k(i) \\ 0 & \text{otherwise,} \end{cases}$$

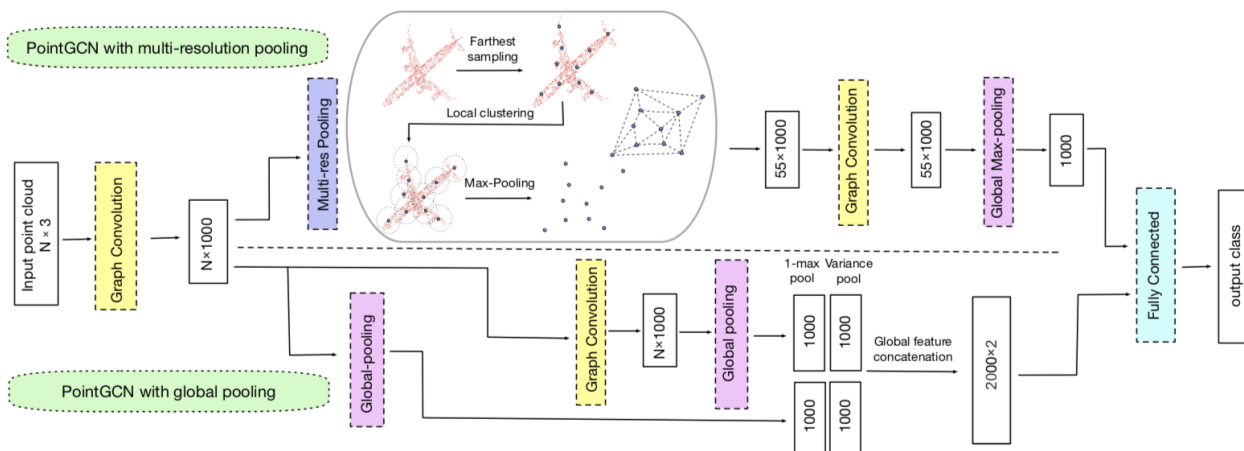
第二步 —— 网络

downsample:



two path: local pooling+global pooling

使用第二代GCN进行卷积运算。



结果

Algorithm	Input format	ModelNet 10 Accuracy (avg. class)	ModelNet 10 Accuracy (overall)	ModelNet 40 Accuracy (avg. class)	ModelNet 40 Accuracy (overall)
3D ShapeNets	volume (1 view)	83.5%	-	77%	84.7%
VoxNet	volume (12 views)	92.0%	-	83%	85.9%
SSCN	volume (20 views)	-	-	88.2%	-
MVCNN	image (80 views)	-	-	90.1%	-
ECC	1024 points (12 views)	90.0%	90.8%	83.2%	87.4 ± 0.40%
PointNet	1024 points (1 view)	91.53%	91.74%	85.35%	88.37 ± 0.33%
PointGCN (global pooling)	1024 points (1 view)	91.39%	91.77%	86.19%	89.27 ± 0.20%
PointGCN (multi-resolution pooling)	1024 points (1 view)	91.57%	91.91%	86.05%	89.51 ± 0.23%
Deep Sets	1000 points (1 view)	-	-	-	87 ± 1%
Deep Sets	5000 points (1 view)	-	-	-	90 ± 0.3%

思考

1. CNN的感受野选取方式极为简单（选取周围节点），却有很好的效果。GCN是否可以借鉴这种naive的感受野选取方式。
2. GCN的感受野选取方式比较make sense（物以类聚），CNN是否可以借鉴这种方式（会增加构图等复杂操作）。
3. GCN构图成功 = 成功一半？
4. 对于3D数据，mesh是不是最合适的构图方式（涉及图形学）。
5. 事先构图依赖于vertices初始的空间位置（点云）或者low level特征之间的相似度（像素），可学习的构图方式？或者维护两张图，一张事先构好，另外一张可学习？
6. 在cv应用中，GCN可以作为后处理，对学习到的特征做聚合。