

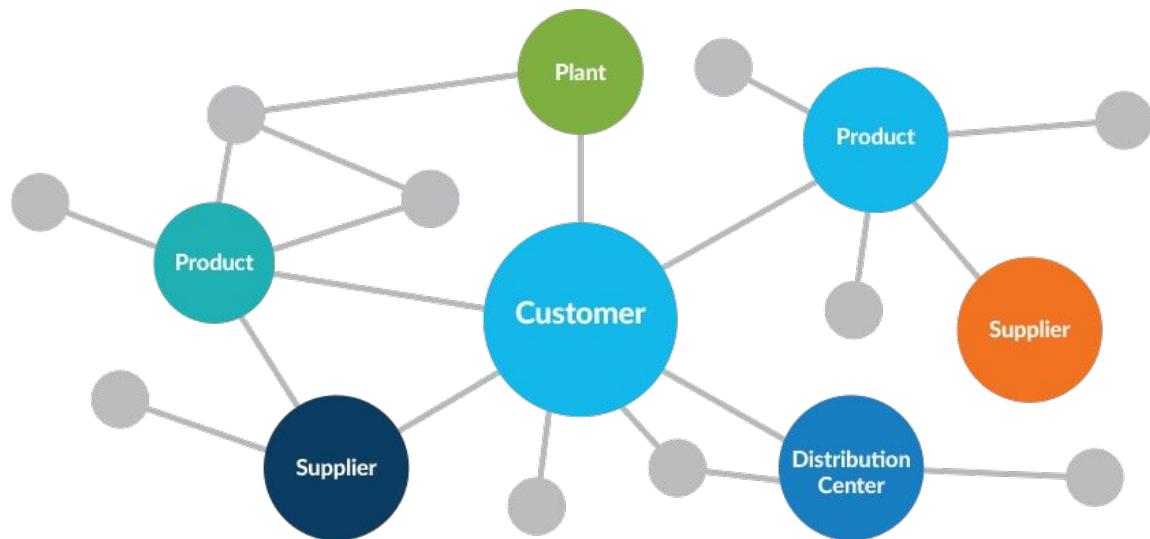
Semi-Supervised Classification with Graph Convolutional Networks

Подготовил: Шимоволос Станислав, 715
13.05.2019

Задача: классификация узлов в графе, где метки доступны только для небольшого числа узлов.

Examples:

- Social networks
- Chemical/Biological Graphs
- Knowledge graphs



Graph convolutional network (GCN) - свёрточная нейронная сеть, работающая с графиками.

Пусть дан граф $G=(V, E)$

На вход сети подаются:

- Матрица признаков $X \in R^{N \times D}$ (N - число вершин, D - число признаков)
- Матрица смежности $A \in R^{N \times N}$
- На выходе каждого слоя получаем матрицу $H^{l+1} \in R^{N \times F}$.
- Каждый слой сети может быть описан как некоторая функция $H^{l+1} = F(H^{(l)}, A)$

Необходимо подобрать и параметризовать функцию F .

Некоторые обозначения

- D (*degree matrix*) – диагональная матрица, на диагонали находятся суммы строк матрицы смежности A (степени вершин графа).
- $\tilde{A} = A + I_N$ - матрица смежности с self-loops
- L (*Graph Laplasian*) – симметрическая матрица, $L = D - A$.
- L_{sym} (нормированный Graph Laplasian) $L_{sym} = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^\top$
- Преобразование Фурье сигнала \mathbf{x} (размерности N) : $\mathcal{F}(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$
- Обратное преобразование Фурье : $\mathcal{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$
- Свёртка сигнала и фильтра : $\mathbf{x} *_G \mathbf{g} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g})) = \mathbf{U}(\mathbf{U}^T \mathbf{x} \odot \mathbf{U}^T \mathbf{g})$

Получим формулу из общих суждений

- Необходимо учесть соседей
- Необходимо учесть собственные признаки

Покажем, что она имеет вид $H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$, $H \in R^{N \times C}$, $W \in R^{C \times F}$ C – число признаков, F – число фильтров

Запишем формулу для “для агрегированной вершины” :

$$\begin{aligned} aggregate(\mathbf{A}, \mathbf{X})_i &= \mathbf{D}^{-0.5} \mathbf{A}_i \mathbf{D}^{-0.5} \mathbf{X} \\ &= \sum_{k=1}^N D_{i,k}^{-0.5} \sum_{j=1}^N A_{i,j} \sum_{l=1}^N D_{j,l}^{-0.5} \mathbf{X}_j \\ &= \sum_{j=1}^N D_{i,i}^{-0.5} A_{i,j} D_{j,j}^{-0.5} \mathbf{X}_j \\ &= \sum_{j=1}^N \frac{1}{D_{i,i}^{0.5}} A_{i,j} \frac{1}{D_{j,j}^{0.5}} \mathbf{X}_j \end{aligned}$$

При вычислении “агрегированного” представления вершины мы учитываем степень i-ой и j-ой вершин :

D_i – нормируем вклад от соседних вершин. В противном случае вершины с большей степенью получали бы большие значения

D_j – увеличиваем вклад соседей с малой степенью вершины и уменьшаем с большой степенью.

Осталось добавить self-loops, домножить на матрицу весов и применить функцию активации.

Далее покажем, что эту формулу можно получить из аппроксимации свёртки сигнала с фильтром на граfe

Некоторые дополнения

- Рассмотрим преобразование Фурье : $\mathcal{F}(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$, его можно интерпретировать как разложение вектора по ортонормированному базису, состоящему из столбцов матрицы $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{n-1}] \in \mathbf{R}^{N \times N}$
- Обратное преобразование Фурье $\mathbf{x} = \sum_i \hat{\mathbf{x}}_i \mathbf{u}_i$, можно рассматривать как восстановление исходного сигнала по его координатам в Фурье базисе
- Можно переписать свёртку несколько в ином виде : $\mathbf{x} *_{\mathcal{G}} \mathbf{g}_{\theta} = \mathbf{U} \mathbf{g}_{\theta} \mathbf{U}^T \mathbf{x} \cdot \mathbf{g}_{\theta} = \text{diag}(\mathbf{U}^T \mathbf{g})$
- Многочлен Чебышева первого рода $T_n(x)$ характеризуется как полином степени n ,

$$\begin{aligned} \text{задаваемый рекуррентно: } T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) & T_0(x) &= 1 \\ && T_1(x) &= x \end{aligned}$$

Замечание

Операция свёртки $\mathbf{g}_{\theta} * \mathbf{x} = \mathbf{U} \mathbf{g}_{\theta} \mathbf{U}^T \mathbf{x}$ является довольно тяжёлой. Как нахождение собственных векторов матрицы L для построения матрицы U , так и умножение на матрицу U ($O(N^2)$) являются довольно дорогими операциями.

- Можно аппроксимировать фильтр с помощью полиномов Чебышева ([2] Hammond 2011) :

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^K \theta'_k T_k(\tilde{\Lambda}), \quad \tilde{\Lambda} = \frac{2}{\lambda_{\max}} \Lambda - I_N \quad \theta' \in \mathbb{R}^K \text{ - вектор коэффициентов полинома.}$$

- Теперь свёртку можно записать в виде (ChebNet) :

$$g_{\theta'} \star x \approx \sum_{k=0}^K \theta'_k T_k(\tilde{L})x, \quad \tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

- Следующий шаг – примем $K = 1$ и $\lambda_{\max} \approx 2$ (GCN) :

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

- Далее для уменьшения числа параметров и для борьбы с переобучением полагаем

$$\theta = \theta'_0 = -\theta'_1 \quad g_{\theta} \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

- И, наконец, делаем *renormalization trick* (уменьшает собственные числа матрицы):

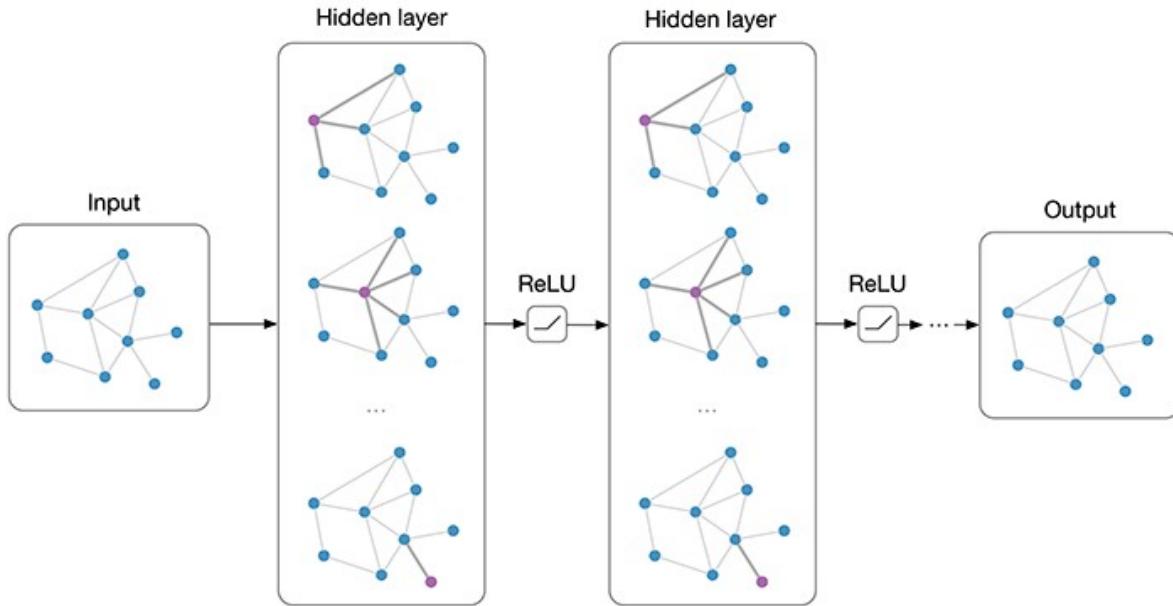
$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

- После обобщения \mathbf{X} на многомерный случай, имеем:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta, \quad \Theta \in \mathbb{R}^{C \times F}, \quad Z \in \mathbb{R}^{N \times F} \quad (C - \text{число признаков}, F - \text{число фильтров})$$

Сложность алгоритма $\mathcal{O}(|\mathcal{E}|FC)$

Переход к нейросети



Прямой проход

$$Z = f(X, A) = \text{softmax}\left(\hat{A} \text{ReLU}\left(\hat{A}XW^{(0)}\right) W^{(1)}\right)$$
$$W^{(0)} \in \mathbb{R}^{C \times H} \quad \hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$
$$W^{(1)} \in \mathbb{R}^{H \times F}$$

Функция потерь (кросс-энтропия на множестве размеченных вершин)

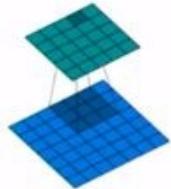
$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

Далее используем градиентный спуск для обучения

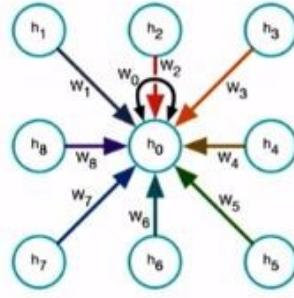
Почему же она “свёрточная”?

Convolutional neural networks on grids

Single CNN layer with 3x3 filter:



(Animation by
Vincent Dumaine)



Update for a single pixel:

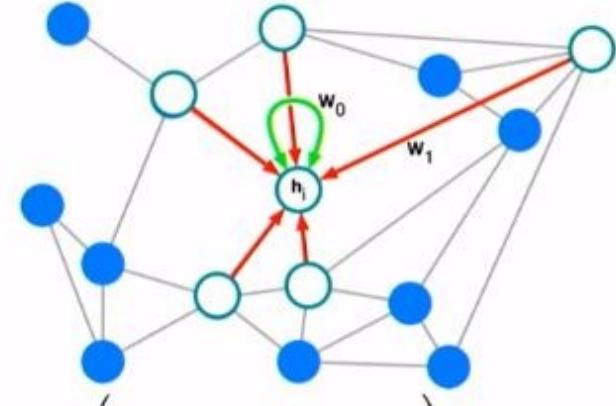
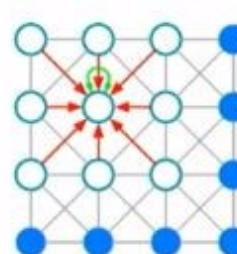
- Transform neighbours individually $\mathbf{W}_i^{(l)} \mathbf{h}_i^{(l)}$
- Add everything up $\sum_i \mathbf{W}_i^{(l)} \mathbf{h}_i^{(l)}$
- Apply nonlinearity $\mathbf{h}_0^{(l+1)} = \sigma \left(\sum_i \mathbf{W}_i^{(l)} \mathbf{h}_i^{(l)} \right)$

<https://aisc.a-i.science/events/2019-03-27/>

Spatial filters for irregular graphs

Related idea was first proposed in Scarselli et al. 2009

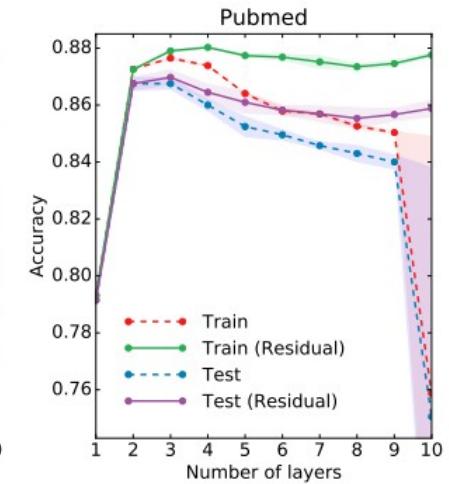
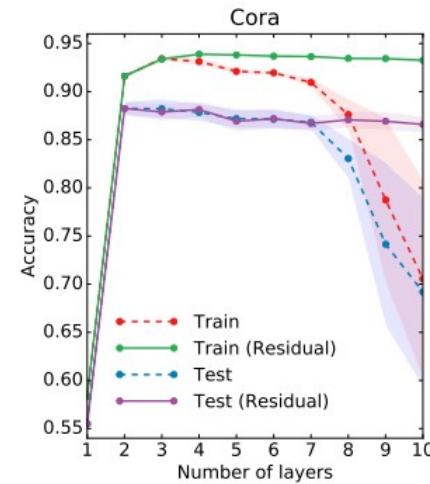
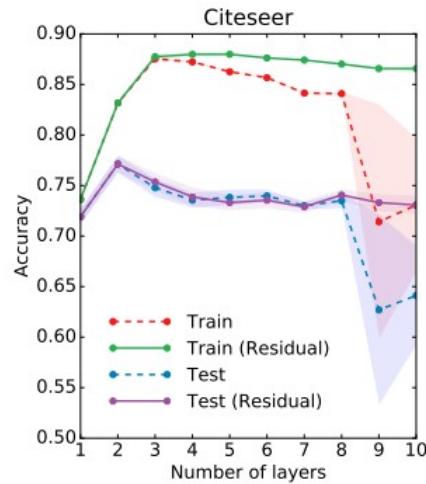
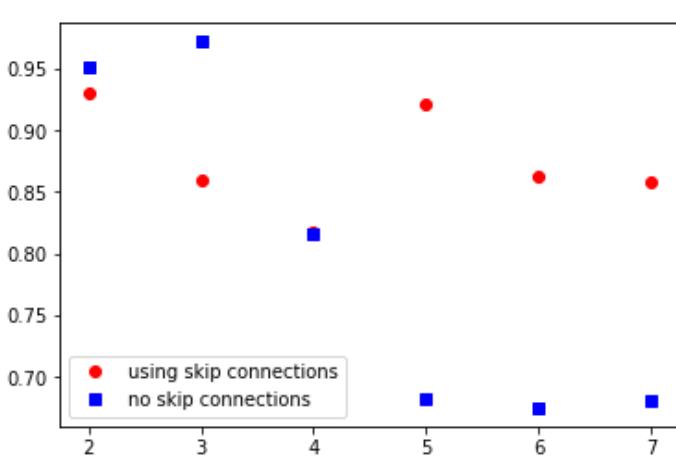
Kipf & Welling ICLR 2017



Propagation rule: $\mathbf{h}_i^{(l+1)} = \sigma \left(\mathbf{h}_i^{(l)} \mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)} \right)$

<https://aisc.a-i.science/events/2019-03-27/>

Эксперименты с числом слоёв



Результаты работы сети github.com

Skip connection

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) + H^{(l)}$$

Список литературы

1. Thomas N. Kipf, Thomas N. Kipf : Semi-Supervised Classification with Graph Convolutional Networks)
2. David K Hammond, Pierre Vandergheynst, and Remi Gribonvalc: Wavelets on Graphs via Spectral Graph Theory
3. Ziwei Zhang, Peng Cui and Wenwu Zhu: Deep Learning on Graphs: A Survey
4. Zonghan Wu, Shirui Pan, Fengwen Chen,
Guodong Long, Chengqi Zhang, Philip S. Yu: A Comprehensive Survey on Graph Neural Networks
5. Felix Wu, Tianyi Zhang, Christopher Fifty, Tao Yu, Kilian Q. Weinberger: Simplifying Graph Convolutional Networks

Спасибо за внимание