FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling

Jie Chen, Tengfei Ma, Cao Xiao. IBM Research

GCN (Graph Convolutional Net)

Graph convolution is similar to image convolution: Sum up the contributions of the neighbors to update the representation of itself.

Properties:
- In standard learning theory, we learn the parameters of a model by minimizing the risk.
- For large models, the minimization is typically done through stochastic gradient descent: Update the model with one training example (or a mini-batch of examples) each time:
  \[ w_{k+1} = w_k - \gamma_k \nabla f(w_k, \xi_k) \]
- The problem for graphs is that one node is related to many other nodes, hence the sample gradient is very expensive to compute.

Empirical Risk Minimization

- Let the graph G have a vertex set \( V \), with an associated probability space \( (V', F, P) \).
- ... such that \( G \) is induced from \( G' \) and the nodes in \( V \) are iid samples of \( V' \) according to probability measure \( P \).
- Now, one layer of GCN...
  \[ H^{(i+1)} = \sigma \left( \hat{A} H^{(i)} W^{(i)} \right) \]
- ... becomes
  \[ h^{(i+1)}(v) = \sigma \left( \int \hat{A}(v, u) h^{(i)}(u) W^{(i)} dP(u) \right) \]

Generalize to Integral Transform

- Set the graph G have a vertex set \( V \), with an associated probability space \( (V', F, P) \).
- We generalize it to an infinite graph \( G' \) with vertex set \( V' \), and an associated probability space \( (V', F, P) \).
- ... such that \( G \) is induced from \( G' \) and the nodes in \( V \) are iid samples of \( V' \) according to probability measure \( P \).
- Now, one layer of GCN...
  \[ H^{(i+1)} = \sigma \left( \hat{A} H^{(i)} W^{(i)} \right) \]
- ... becomes
  \[ h^{(i+1)}(v) = \sigma \left( \int \hat{A}(v, u) h^{(i)}(u) W^{(i)} dP(u) \right) \]

Theoretical Result

- Recall that we approximate the loss function \( f \) by sampling the \( l \)-th layer with \( t \) nodes:
  \[ f = \mathbb{E}_{u \sim P}[f(w; h^{(M)}(v))] \approx f_{t_0, t_1, \ldots, t_M} = \frac{1}{M} \sum_{i=1}^{M} f(w; h_{t_0, t_1, \ldots, t_M}^{(i)}) \]
- Theorem: The approximator is strongly consistent:
  \[ \lim_{t_0, t_1, \ldots, t_M \to \infty} f_{t_0, t_1, \ldots, t_M} = f \text{ with probability one} \]
- The result may be easily generalized to the gradient:
  \[ \lim_{t_0, t_1, \ldots, t_M \to \infty} \nabla f_{t_0, t_1, \ldots, t_M} = \nabla f \text{ with probability one} \]

Experimental Results

- Optimizing the implementation of GraphSAGE for small graphs improves running time to here.

Experimental Results

- Table 4: Total training time (in seconds).

Sampling Each Layer

- We may perform Monte Carlo approximation for the integral in each layer. For the \( l \)-th layer, use \( t_l \) iid samples \( \sim P \):
  \[ h^{(i+1)}(v) = \sigma \left( \frac{1}{t_l} \sum_{i=1}^{t_l} \hat{A}(v, u^{(i)}) h^{(i)}(u^{(i)}) W^{(i)} \right) \]
- In practice, this sampling means using \( t_l \) iid nodes \( \sim \) Uniform to approximate matrix multiplication (because vertex set is already iid \( \sim P \))
  \[ H^{(i+1)}(v, \cdot) = \sigma \left( \frac{1}{t_l} \sum_{i=1}^{t_l} \hat{A}(v, u^{(i)}) H^{(i)}(u^{(i)}, \cdot) W^{(i)} \right) \]
- Furthermore, for variance reduction, we may use \( t_l \) iid nodes \( \sim \) Q to perform sampling. The optimal distribution \( Q \) is not efficient to compute, but setting \( Q \) to be proportional to the squared column norms of the normalized adjacency matrix \( \hat{A} \) works fairly well in practice.