Loss Function for Skip-Gram methods (III)

Yitao (Viola) Chen

Noise Contrastive Estimation (NCE) / Negative sampling

- so far, only maximizing probability of word/node within context
- here, aims at maximizing the similarity of the words in the same context and minimizing it when they occur in different contexts
- idea: get a noise distribution by negative sampling from unrelated parts of the graph

Some notation in graphs

- k-step transition matrix: A^k
  (why?)
- Pr(go from node v_i to node v_j in k steps) $= A^k_{i,j}$,
  let this be denoted by $p_k(v_j|v_i)$
- let $p_k(V)$ be some distribution over vertices in the graph
  (I actually couldn't find more details on this)
- sample a vertex c according to $p_k(V)$ – negative sampling, then
  $p_k(c) = \frac{1}{N}\sum_{v_i} A^k_{v_i,c}$

Loss function

- λ: hyperparameter indicating no. of negative samples
- define local loss over a specific pair (v_i, v_j):

$$L_k(v_i, v_j) = p_k(v_j|v_i) \cdot \log(\sigma(\Phi(v_i) \cdot \Phi(v_j)) + \lambda \cdot p_k(v_j) \cdot \log \sigma(-\Phi(v_i) \cdot \Phi(v_j))$$

$$= A^k_{v_i,v_j} \cdot \log(\sigma(\Phi(v_i) \cdot \Phi(v_j)) + \lambda \cdot p_k(v_j) \cdot \log \sigma(-\Phi(v_i) \cdot \Phi(v_j))$$

- k-step loss function of a node:
  $L_k(v_i) = \sum_{v_j \neq v_i} L_k(v_i, v_j)$
- k-step loss function over whole graph
  $L_k = \sum_{v \in V} L_k(v)$

Loss function

- λ: hyperparameter indicating no. of negative samples
- define local loss over a specific pair (v_i, v_j):

$$
\begin{aligned}
L_k(v_i, v_j) &= p_k(v_j|v_i) \cdot \log(\sigma(\Phi(v_i) \cdot \Phi(v_j)) + \lambda \cdot p_k(v_j) \cdot \log \sigma(-\Phi(v_i) \cdot \Phi(v_j)) \\
&= A^k_{v_i, v_j} \cdot \log(\sigma(\Phi(v_i) \cdot \Phi(v_j)) + \lambda \cdot p_k(v_j) \cdot \log \sigma(-\Phi(v_i) \cdot \Phi(v_j))
\end{aligned}
$$

- k-step loss function of a node:
  $L_k(v_i) = \sum_{v_j \neq v_i} L_k(v_i, v_j)$
- k-step loss function over whole graph
  $L_k = \sum_{v \in V} L_k(v)$

Back propagation

- let $z = \Phi(v_i) \cdot \Phi(v_j)$, and set $\dfrac{\delta L_k}{\delta z} = 0$
  we get
  $$\Phi(v_i) \cdot \Phi(v_j) = \log\left(\frac{A^k_{v_i,v_j}}{\sum_w A^k_{w,v_j}}\right) - \log\frac{\lambda}{N}$$
  Let this matrix be Y^k
  (I didn't do the math...perhaps you can try)
- For SVD, we want at least a semi positive-definite matrix, so we remove the negative values in Y^k by letting $X^k = max(Y^k, 0)$,
  so $X^k_{i,j} \approx \Phi(v_i) \cdot \Phi(v_j)$ after replacing the negative entries with 0
- X^k can then be factored with SVD or other methods to get $X^k \approx U\Sigma V$
- thus $\Phi \approx U(\Sigma)^{\frac{1}{2}}$

GraRep - Algorithm

GraRep Algorithm

Input
Adjacency matrix S on graph
Maximum transition step K
Log shifted factor β
Dimension of representation vector d

1. Get k-step transition probability matrix A^k
Compute $A = D^{-1}S$
Calculate $A^1, A^2, \ldots, A^K$, respectively
2. Get each k-step representations
For $k = 1$ to K
  2.1 Get positive log probability matrix
  calculate $\Gamma_1^k, \Gamma_2^k, \ldots, \Gamma_N^k$ ($\Gamma_j^k = \sum_p A_{p,j}^k$) respectively
  calculate $\{X_{i,j}^k\}$

$$X_{i,j}^k = \log\left(\frac{A_{i,j}^k}{\Gamma_j^k}\right) - \log(\beta)$$

  assign negative entries of X^k to 0
  2.2 Construct the representation vector W^k
  $[U^k, \Sigma^k, (V^k)^T] = SVD(X^k)$
  $W^k = U_d^k(\Sigma_d^k)^{\frac{1}{2}}$
End for
3. Concatenate all the k-step representations
$W = [W^1, W^2, \ldots, W^K]$

Output
Matrix of the graph representation W

Other ideas

- for vertex-vertex relationship, it doesn't have to be dot-product
- other methods include but not limited to
  - common neighbors $|N(u) \cap N(v)|$
  - Jaccard's coefficient $\dfrac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$
  - Adamic-Adar score $\sum_{t \in |N(u) \cap N(v)|} \dfrac{1}{\log |N(t)|}$
  - Preferential attachment $|N(u)| \cdot |N(v)|$

  for vertex pair u, v with neighbor set $N(u)$ and $N(v)$

Summary

- random walk is a way to capture local (and by induction, global) structure of a graph
- this class of methods is non-Euclidean, and loss function is different
- suitable for scenarios where the underlying graphical structure is known, and we want to be able to learn node representation or infer nodes in close neighborhood with ease in a particularly large graph

References

- Distributed Large-scale Natural Graph Factorization.
  Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy.
- GraRep: Learning Graph Representations with Global Structural information.
  Shaosheng Cao, Wei Lu, Qiongkai Xu.
- DeepWalk: Online Learning of Social Representations.
  Bryan Perozzi, Rami Al-Rfou, Steven Skiena
- Distributed Representations of Words and Phrases and their compositionality
  Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean

Thank you for listening!

Yitao Chen

chen_yitao@gis.a-star.edu.sg