Multi-label Classification on Networked Data

Han Xiao

Department of Computer Science
Aalto University

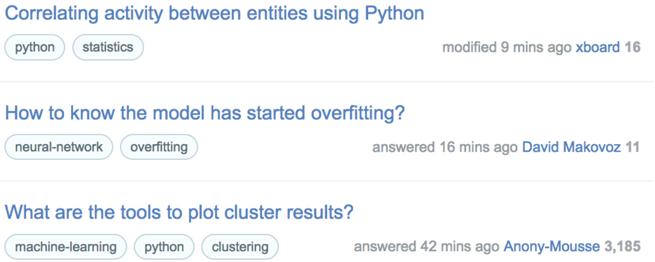
September 2017

overview

- **goal**: multi-label classification of posts on QA-sites
- **ingredient 1**: deep learning for supervised learning
- **ingredient 2**: graph embedding learning for under supervised learning
- both requires convex optimization techniques

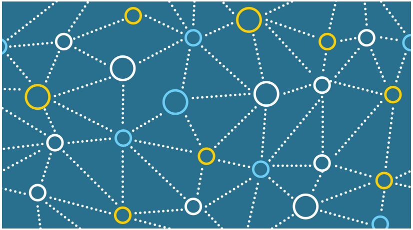
motivation 1

- goal: learn a automatic question tagger
- example: stackoverflow questions
- tags: python, java, android, ios, ...



motivation 2

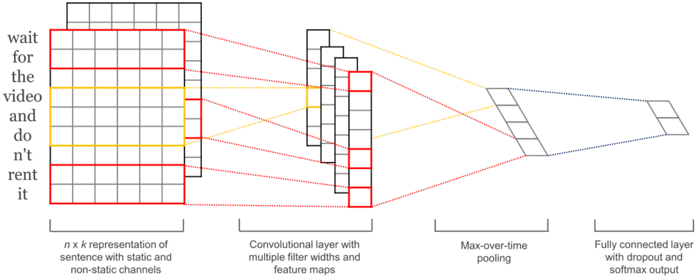
- posts and users are linked in a network
  - user *post/likes/comment* posts
- can we leverage *network information* for tagging purpose?



multi-label classification

- given n pairs of feature vectors and label vectors, $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0, 1\}^L$
- learn a function $f : \mathbb{R}^D \to \{0, 1\}^L$ that minimizes some error function
- our problem can be modeled as such

multi-label classification: CNN-based approach



[1]

- loss function: softmax is replaced by sigmoid

[1] Kim, Yoon. "Convolutional neural networks for sentence classification." EMNLP (2014).

graph embedding

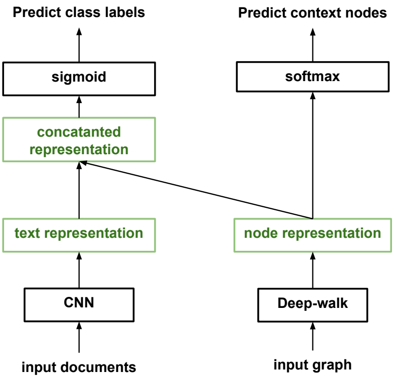
- given $G = (V, E)$, $V = \{1, \ldots, n\}, E \subseteq V \times V$
- learn a function $f : V \to \mathbb{R}^d$
- $f(v)$ is the embedding for node v
- demonstrated to improve clustering, classification performance in general

example: DeepWalk [2]

- idea borrowed from word2vec (a word predicts its context/surrounding word)
- here, think of a node as a word
- a node predicts its “context” nodes
- context nodes are collected by random walk

[2] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. “Deepwalk: Online learning of social representations.” KDD, 2014.

model & architecture



training objective: $\mathcal{L} = \mathcal{L}_s + \mathcal{L}_u$

model: unsupervised term $\mathcal{L}_u$

- essentially deepwalk
- given $\{(i, c)\}$ pairs, learn a model that miminimizes:

$$\begin{aligned}\mathcal{L}_u &= -\mathrm{E}_{(i,c)} \log \Pr(\mathbf{e}_c \mid \mathbf{e}_i) \\ &= -\mathrm{E}_{(i,c)} \log \frac{\mathbf{e}_c^T \mathbf{e}_i}{\sum_{c'} \mathbf{e}_{c'}^T \mathbf{e}_i}\end{aligned}$$

- (i, c) pairs collected from random walk

model: supervised term $\mathcal{L}_s$

- given $\{(\mathbf{x}_i, \mathbf{e}_i, \mathbf{y}_i)\}_{i=1}^{L}$ triplets, learn a model that miminimizes:

$$\mathcal{L}_s = \frac{1}{L} \sum_{i=1}^{K} \log p(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{e}_i)$$

- model similar to KimCNN (using sigmoid instead of softmax)

training

- alternating update of $\mathcal{L}_u$ and $\mathcal{L}_s$
- optimizer: ADAM

experiment: datasets & evaluation

- 3 sites from stackexchange.com:

	data science	**emacs**	**software engineering**
#instances	5145	4536	10336
#labels	327	618	1344
avg #labels per	2.7	2.0	2.6

- question network contruction: two questions are linked they are linked to at least one common user
- train/dev/test ratio: 80%/10%/10%
- evaluation: precision@k, $k = \{1, 3, 5\}$

methods

- *fastxml*: tf-idf features
- *cnn*: sigmoid loss function
- *cnn* + *deepwalk*

results: data science

	fastxml	*cnn*	*cnn+deepwalk*
p@1	0.26	0.53	**0.57**
p@3	0.17	0.37	**0.38**
p@5	0.15	**0.29**	0.28

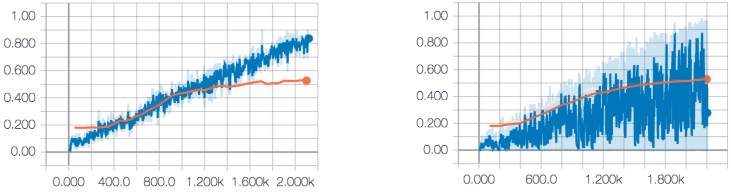
results: emacs

	fastxml	*cnn*	*cnn+deepwalk*
p@1	0.21	**0.58**	0.56
p@3	0.10	0.30	0.30
p@5	0.07	0.20	0.20

results: software engineering

	fastxml	*cnn*	*cnn+deepwalk*
p@1	0.06	0.41	**0.43**
p@3	0.06	**0.26**	0.25
p@5	0.05	0.18	0.18

train/dev curve



(a) cnn

(b) cnn+deepwalk

Figure: percision@1 for *emacs*

conclusion

- I'm suprised at CNN's performance
- also suprised that network embedding does not help much
- possible reasons: improper network construction, formulation, training, etc

future plan

- try different ways of learning network embedding
- understand how graph embedding can help
- design scalable methods for classification (especially when label space is large)