Notes on:
Planetoid: Revisiting Semi-Supervised Learning with Graph Embeddings

September 3, 2024

1 Abstract

Given a graph between instances, embeddings for each instance to jointly predict the class label and the neighborhood context in the graph was trained. Transductive and inductive variants of the method were developed.

2 Transductive and Inductive methods

Transductive method makes predictions based on embeddings specific to the input data and uses these embeddings in conjunction with the input feature vectors to predict class labels. This method leverages the relationship in the training data, including unlabeled data, to make predictions for a specific test set.

Inductive method defines embeddings as a parametric function of the feature vectors, enabling predictions on instances, not seen during training by generalizing the learned function. This method relies on the input features and a learned model that can apply to any instance, not just those present during training.

3 Introduction

Planetoid (Predicting Labels And Neighbors with Embeddings Transductively or Inductively from Data) main contribution is to incorporate embedding techniques into graph-based semi-supervised learning setting.

Semi-supervised learning aims to leverage unlabeled data to improve performance. In this paper graph embeddings were considered instead of word embeddings. Since embeddings are learned based on the graph structure, this method is transductive, which means it can only predict instances that are already observed in the graph at training time.

An inductive variant is also presented where the embeddings are defined as a parameterized function of input feature vectors; i.e., the embeddings can be viewed as hidden layers of a neural network.

4 Semi-Supervised Learning

Let L and U be the the labeled and unlabeled sets. Let $X_{1:L}$ and $X_{L+1:L+U}$ denote the feature vectors of labeled and unlabeled instances respectively. Semi-supervised learning can be defined as a classifier $f : x \to y$.

There are two learning paradigms. Transudctive and Inductive learning. Transductive learning only aims to apply the classifier f on the unlabeled instances observed at training time, and the classifier does not generalise to unobserved instances. Inductive learning, on the other hand, aims to learn a parameterized classifier f that is generalizable to unobserved instances.

5 Graph-Based Semi-Supervised Learning

Graph-based semi-supervised learning is based on the assumption that nearby nodes tend to have the same labels. Generally, the loss function of graph-based semi-supervised learning in the binary case can be written as:

$$\sum_{i=1}^{L} l\left(y_i, f(x_i)\right) + \lambda \sum_{i,j} a_{ij} \|f(x_i) - f(x_j)\|^2 \tag{1}$$

$$= \sum_{i=1}^{L} l\left(y_i, f(x_i)\right) + \lambda \mathbf{f}^T \Delta \mathbf{f} \tag{2}$$

The first term is the standard supervised loss function ($l(\cdot,\cdot)$). The second term is the graph Laplacian regularization, which incurs a large penalty when similar nodes with a large w_{ij} are predicted to have different labels $f(x_i) \neq f(x_j)$. The graph Laplacian matrix Δ(often denoted as L) is defined as $\Delta = A - D$, where D is a diagonal matrix with each entry defined as $d_{ii} = \sum_j a_{ij}$. This means that the diagonal elements of D are the degree of vertex of i (number of edges connected to vertex i).

6 Semi-Supervised Learning with Graph Embeddings

This framework is formulated based on feed-forward neural networks. Given the input feature vector x, the k-th hidden layer of the network is denoted as $\mathbf{h}^k$, which is a nonlinear function of the previous hidden layer h^{k-1} defined as: $h^k(x) = \text{ReLU}(\mathbf{W}^k \mathbf{h}^{k-1} + b^k)$, where $\mathbf{W}^k$ and b^k are parameters of the k-th layer, and $\mathbf{h}^0(x) = x$.

The loss function the network can be expressed as $\mathcal{L}+\lambda\mathcal{L}_u$, where $\mathcal{L}_s$ is a supervised loss of predicting the labels and $\mathcal{L}_u$ the unsupervised loss of predicting the graph context.

Negative sampling is introduced in this work to approximate the normalization term. In this case, sampling (i, c, γ) from distribution, where i and c denote instance and context respectively, γ =1 or -1 if its positve or negative pair, respectively.

Given (i, c, γ), cross-entropy of classifying the pair (i, c) is minimised to a binary label γ:

$$\mathbb{I}(\gamma = 1) \log \sigma(\mathbf{w}_c^T \mathbf{e}_i) - \mathbb{I}(\gamma = -1) \log \sigma(-\mathbf{w}_c^T \mathbf{e}_i) \tag{3}$$

where σ is the sigmoid function, and $\mathbb{I}$ is a binary indicator that outputs 1 when the argument is true otherwise 0. Therefore, the unsupervised loss with negative sampling function can be written as:

$$\mathcal{L} = -\mathbb{E}_{i,c,\gamma} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i) \tag{4}$$

7 Transductive Formulation

This method infers the labels of unlabeled instances $y_{L+1:L+U}$ without generalizing unobserved instances. k layers are applied on the input feature vector x to obtain $\mathbf{h}^k(x)$, and l on the embedding $\mathbf{e}$ to obtain $\mathbf{h}^l(e)$. The two hidden layers are concatenated, and fed to a softmax layer to predict class label of the instance. More specifically, the probability of predicting the label y is written as:

$$p(y \mid \mathbf{x}, \mathbf{e}) = \frac{\exp[\mathbf{h}^k(x)^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(x)^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_{y'}} \tag{5}$$

where $[\cdot, \cdot]$ denotes concatenation of two row vectors and $\mathbf{w}$ represents the model parameter.

Combining with Eq. 4, the loss function of transductive learning is defined as:

$$-\frac{1}{L} \sum_{i=1}^{L} \log p(y_i \mid \mathbf{x}_i, \mathbf{e}_i) - \lambda \mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i) \tag{6}$$

This formulation is transductive because the prediction of label y depends on the embeddings $\mathbf{e}$, which can only be learned for instances observed in graph A during training time.

8 Inductive Formulation

To make this method inductive, the prediction of label y should only depend on the input feature x. Therefore, the embedding $\mathbf{e}$ is defined as a parameterized function of feature x.

k layers are applied on the input feature vector x to obtain $\mathbf{h}^k(x)$. However, rather than using a "free embedding", l_1 layers is applied on the input feature vector and define it as the embedding $\mathbf{e} = \mathbf{h}^{l_1}(x)$. Then another layer l_2 are applied on the embedding $\mathbf{h}^{l_2}(\mathbf{e}) = \mathbf{h}^{l_2}(\mathbf{h}^{l_1})$, denoted as $\mathbf{h}^l(x)$.

Label y only depends on feature x. More specifically:

$$p(y \mid \mathbf{x}) = \frac{\exp[\mathbf{h}^k(x)^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(x)^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_{y'}} \tag{7}$$

Now, to generalise the loss function, Eq. 4 $\mathbf{e}_i$ can be replaced with $\mathbf{h}^{l_1}$.