

PEG-PYG Description

Haorui Wang

Mar 2022

The PEG layer from the “[Equivariant and Stable Positional Encoding for More Powerful Graph Neural Networks](#)”

The PEG layer:

$$X', Z' = (\sigma[(\hat{A} \odot M)XW], Z)$$

where $M_{uv} = MLP(\|Z_u - Z_v\|), \forall u, v \in V$. $\hat{A} = \hat{D}^{-1/2}(A + I)\hat{D}^{-1/2}$ is the normalized adjacent matrix and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ is diagonal degree matrix. \odot denotes Hadamard product and Z is the positional encoding. The adjacency matrix can include other values than 1 representing edge weights via the optional edge_weight tensor.

PARAMETERS:

- **in_feats_dim:** (int) Size of each input node feature sample
- **pos_dim:** (int) Size of each input positional encoding sample. Notice in PEG we do not update the positional encodings.
- **out_feats_dim:** (int) Size of each output node embedding sample.
- **improved:** (bool, optional) If set to :obj:‘True’, the layer computes \hat{A} as $A + 2I$. (default: ‘False’)
- **cached:** (bool, optional) If set to: True, the layer will cache the computation of $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ on first execution, and will use the cached version for further executions. This parameter should only be set to: ‘True’ in transductive learning scenarios. (default: ‘False’)
- **add_self_loops:** (bool, optional) If set to: ‘False’, will not add self-loops to the input graph. (default: ‘True’)
- **normalize:** (bool, optional) Whether to add self-loops and compute symmetric normalization coefficients on the fly. (default: ‘True’)
- **bias:** (bool, optional) If set to: ‘False’, the layer will not learn an additive bias. (default: ‘True’)

- **update_coors:** (bool, optional) Whether to update positional encodings. (default: 'False')
- **use_formerinfo:** (bool, false) Whether to use previous layer's output to update node features. (default: 'False')
- **norm_coors:** Whether to normalize positional encodings. Only used when `update_coors = True`. (default: 'False')
- ****kwargs:** (optional) Additional arguments of : class:'torch_geometric.nn.conv.MessagePassing'.

SHAPES

- **input:** node features : $(|\mathcal{V}|, F_{in})$, positional encodings: $(|\mathcal{V}|, P_{in})$, edge indices: $(2, |\mathcal{E}|)$, edge weights: $(|\mathcal{E}|)$ (optional)
- **output:** node features: $(|\mathcal{V}|, F_{out})$, positional encodings: $(|\mathcal{V}|, P_{out})$

`reset_parameters()`

`forward(x: torch.Tensor, edge_index: Union[torch.Tensor, torch_sparse.tensor.SparseTensor], edge_weight: Optional[torch.Tensor] = None) → torch.Tensor`