PEGConv Description

Haorui Wang

April 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 $\hat{D}_{ii} = \Sigma_{j=0}\hat{A}_{ij}$ 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_channels:** (int) Size of each input node feature sample
- **out_channels:** (int) Size of each output node embedding sample.
- **edge_mlp_dim:** (int) We use MLP to make one to one mapping between the relative information and edge weight. edge_mlp_dim represents the hidden units dimension in the MLP. (default: 32)
- **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‘)

- ****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})

reset_parameters()

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