JAXChem

Release 0.0.1.dev0

deepchem-contributors

Jul 08, 2020

NOTES

JAXChem is a JAX-based deep learning library for complex and versatile chemical modelings.

CHAPTER

ONE

INSTALLATION

1.1 pip installation

JAXChem requires the following packages.

- JAX (jax==0.1.69, jaxlib==0.1.47)
- Haiku (==0.0.1)
- typing-extensions (>=3.7.4)

First, you have to install JAX. Please confirm how to install JAX from here.
After installing JAX, please run the following commands.

```
// install jaxchem
$ pip install git+https://github.com/deepchem/jaxchem
```

1.2 docker installation

Please run the following commands.

```
$ git clone https://github.com/deepchem/jaxchem.git
$ cd jaxchem
$ docker build . -t jaxchem
```

CHAPTER

TWO

JAXCHEM.LOSS

binary_cross_entropy_with_logits(*inputs*, *targets*, *average=True*)

Binary cross entropy loss.

This function is based on the PyTorch implemantation.

See : https://discuss.pytorch.org/t/numerical-stability-of-bcewithlogitsloss/8246

Parameters

- **inputs** (*jnp.ndarray*) – This is a model output. This is a value before passing a sigmoid function.
- **targets** (*jnp.ndarray*) – This is a label and the same shape as inputs.
- **average** (*bool*) – Whether to mean loss values or sum, default to be True.

Returns **loss** – This is a binary cross entropy loss.

Return type jnp.ndarray

CHAPTER

THREE

JAXCHEM.MODELS

Contents

3.1 GCN

class PadGCNPredicator(**args*, ***kwargs*)

GCN Predicator is a wrapper function using GCN and MLP.

__init__(*in_feats*, *hidden_feats*, *activation=None*, *batch_norm=None*, *dropout=None*, *pooling_method='mean'*, *predicator_hidden_feats=128*, *predicator_dropout=0.0*, *n_out=1*, *name=None*)

Initializes the module.

Parameters

- **in_feats** (*int*) – Number of input node features.
- **hidden_feats** (*list[int]*) – List of output node features.
- **activation** (*list[Activation] or None*) – `activation[i]` is the activation function of the i-th GCN layer. `len(activation)` equals the number of GCN layers. By default, the activation each layer is relu function.
- **batch_norm** (*list[bool] or None*) – `batch_norm[i]` decides if batch normalization is to be applied on the output of the i-th GCN layer. `len(batch_norm)` equals the number of GCN layers. By default, batch normalization is not applied for all GCN layers.
- **dropout** (*list[float] or None*) – `dropout[i]` decides the dropout probability on the output of the i-th GCN layer. `len(dropout)` equals the number of GCN layers. By default, dropout is not performed for all layers.
- **pooling_method** (*Literal['max', 'min', 'mean', 'sum']*) – pooling method name, default to 'mean'.
- **predicator_hidden_feats** (*int*) – Size of hidden graph representations in the predicator, default to 128.
- **predicator_dropout** (*float*) – The probability for dropout in the predicator, default to 0.0.

- **n_out** (*int*) – Number of the output size, default to 1.
- **name** (*Optional[str]*) –

__call__(*node_feats*, *adj*, *is_training*)
Predict logits or values

Parameters

- **node_feats** (*ndarray of shape (batch_size, N, in_feats)*) – Batch input node features. N is the total number of nodes in the batch of graphs.
- **adj** (*ndarray of shape (batch_size, N, N)*) – Batch adjacency matrix.
- **is_training** (*bool*) – Whether the model is training or not.

Returns **out** – Predicator output.

Return type ndarray of shape (batch_size, n_out)

class PadGCN(**args*, ***kwargs*)
GCN module. Paper: Semi-Supervised Classification with Graph Convolutional Networks

__init__(*in_feats*, *hidden_feats*, *activation=None*, *batch_norm=None*, *dropout=None*, *name=None*)
Initializes the module.

Parameters

- **in_feats** (*int*) – Number of input node features.
- **hidden_feats** (*list[int]*) – List of output node features.
- **activation** (*list[Activation] or None*) – `activation[i]` is the activation function of the i-th GCN layer. `len(activation)` equals the number of GCN layers. By default, the activation each layer is relu function.
- **batch_norm** (*list[bool] or None*) – `batch_norm[i]` decides if batch normalization is to be applied on the output of the i-th GCN layer. `len(batch_norm)` equals the number of GCN layers. By default, batch normalization is not applied for all GCN layers.
- **dropout** (*list[float] or None*) – `dropout[i]` decides the dropout probability on the output of the i-th GCN layer. `len(dropout)` equals the number of GCN layers. By default, dropout is not performed for all layers.
- **name** (*Optional[str]*) –

__call__(*node_feats*, *adj*, *is_training*)
Update node features.

Parameters

- **node_feats** (*ndarray of shape (batch_size, N, in_feats)*) – Batch input node features. N is the total number of nodes in the batch of graphs.
- **adj** (*ndarray of shape (batch_size, N, N)*) – Batch adjacency matrix.
- **is_training** (*bool*) – Whether the model is training or not.

Returns **new_node_feats** – Batch new node features.

Return type ndarray of shape (batch_size, N, out_feats)

class PadGCNLayer(**args*, ***kwargs*)
Single GCN layer from Semi-Supervised Classification with Graph Convolutional Networks

__init__(*in_feats*, *out_feats*, *activation=None*, *bias=True*, *normalize=True*, *batch_norm=False*, *dropout=0.0*, *w_init=None*, *b_init=None*, *name=None*)
Initializes the module.

Parameters

- **in_feats** (*int*) – Number of input node features.
- **out_feats** (*int*) – Number of output node features.
- **activation** (*Activation or None*) – activation function, default to be relu function.
- **bias** (*bool*) – Whether to add bias after affine transformation, default to be True.
- **normalize** (*bool*) – Whether to normalize the adjacency matrix or not, default to be True.
- **batch_norm** (*bool*) – Whetehr to use BatchNormalization or not, default to be False.
- **dropout** (*float*) – The probability for dropout, default to 0.0.
- **W_init** (*initialize function for weight*) – Default to be He truncated normal distribution.
- **b_init** (*initialize function for bias*) – Default to be truncated normal distribution.
- **w_init** (*Optional[Callable[[Sequence[int], Any], jax.numpy.lax_numpy.ndarray]]*) –
- **name** (*Optional[str]*) –

__call__(*node_feats*, *adj*, *is_training*)
Update node features.

Parameters

- **node_feats** (*ndarray of shape (batch_size, N, in_feats)*) – Batch input node features. N is the total number of nodes in the batch of graphs.
- **adj** (*ndarray of shape (batch_size, N, N)*) – Batch adjacency matrix.
- **is_training** (*bool*) – Whether the model is training or not.

Returns **new_node_feats** – Batch new node features.

Return type ndarray of shape (batch_size, N, out_feats)

class SparseGCNPredicator(**args*, ***kwargs*)
GCN Predicator is a wrapper function using GCN and MLP.

__init__(*in_feats*, *hidden_feats*, *activation=None*, *batch_norm=None*, *dropout=None*, *pooling_method='mean'*, *predicator_hidden_feats=128*, *predicator_dropout=0.0*, *n_out=1*, *name=None*)
Initializes the module.

Parameters

- **in_feats** (*int*) – Number of input node features.
- **hidden_feats** (*list[int]*) – List of output node features.
- **activation** (*list[Activation] or None*) – `activation[i]` is the activation function of the i-th GCN layer. `len(activation)` equals the number of GCN layers. By default, the activation each layer is relu function.

- **batch_norm** (*list[bool] or None*) – batch_norm[i] decides if batch normalization is to be applied on the output of the i-th GCN layer. len(batch_norm) equals the number of GCN layers. By default, batch normalization is not applied for all GCN layers.
- **dropout** (*list[float] or None*) – dropout[i] decides the dropout probability on the output of the i-th GCN layer. len(dropout) equals the number of GCN layers. By default, dropout is not performed for all layers.
- **pooling_method** (*Literal['max', 'min', 'mean', 'sum']*) – pooling method name, default to 'mean'.
- **predicator_hidden_feats** (*int*) – Size of hidden graph representations in the predicator, default to 128.
- **predicator_dropout** (*float*) – The probability for dropout in the predicator, default to 0.0.
- **n_out** (*int*) – Number of the output size, default to 1.
- **name** (*Optional[str]*) –

__call__(*node_feats*, *adj*, *graph_idx*, *is_training*)
Predict logits or values

Parameters

- **node_feats** (*ndarray of shape (N, in_feats)*) – Batch input node features. N is the total number of nodes in the batch
- **adj** (*ndarray of shape (2, E)*) – Batch adjacency list. E is the total number of edges in the batch
- **graph_idx** (*ndarray of shape (N,)*) – This idx indicate a graph number for node_feats in the batch. When the two nodes shows the same graph idx, these belong to the same graph.
- **is_training** (*bool*) – Whether the model is training or not.

Returns **out** – Predicator output.

Return type ndarray of shape (batch_size, n_out)

class SparseGCN(**args*, ***kwargs*)
GCN module. Paper: Semi-Supervised Classification with Graph Convolutional Networks

__init__(*in_feats*, *hidden_feats*, *activation=None*, *batch_norm=None*, *dropout=None*, *name=None*)
Initializes the module.

Parameters

- **in_feats** (*int*) – Number of input node features.
- **hidden_feats** (*list[int]*) – List of output node features.
- **activation** (*list[Activation] or None*) – activation[i] is the activation function of the i-th GCN layer. len(activation) equals the number of GCN layers. By default, the activation each layer is relu function.
- **batch_norm** (*list[bool] or None*) – batch_norm[i] decides if batch normalization is to be applied on the output of the i-th GCN layer. len(batch_norm) equals the number of GCN layers. By default, batch normalization is not applied for all GCN layers.

- **dropout** (*list[float] or None*) – `dropout[i]` decides the dropout probability on the output of the i-th GCN layer. `len(dropout)` equals the number of GCN layers. By default, dropout is not performed for all layers.
- **name** (*Optional[str]*) –

__call__(*node_feats*, *adj*, *is_training*)
Update node features.

Parameters

- **node_feats** (*ndarray of shape (N, in_feats)*) – Batch input node features. N is the total number of nodes in the batch.
- **adj** (*ndarray of shape (2, E)*) – Batch adjacency list. E is the total number of edges in the batch.
- **is_training** (*bool*) – Whether the model is training or not.

Returns **new_node_feats** – Batch new node features.

Return type ndarray of shape (N, out_feats)

class SparseGCNLayer(**args*, ***kwargs*)
Single GCN layer from Semi-Supervised Classification with Graph Convolutional Networks

__init__(*in_feats*, *out_feats*, *activation=None*, *bias=True*, *normalize=True*, *batch_norm=False*, *dropout=0.0*, *w_init=None*, *b_init=None*, *name=None*)
Initializes the module.

Parameters

- **in_feats** (*int*) – Number of input node features.
- **out_feats** (*int*) – Number of output node features.
- **activation** (*Activation or None*) – activation function, default to be relu function.
- **bias** (*bool*) – Whether to add bias after affine transformation, default to be True.
- **normalize** (*bool*) – Whether to normalize or not, default to be True.
- **batch_norm** (*bool*) – Whetehr to use BatchNormalization or not, default to be False.
- **dropout** (*float*) – The probability for dropout, default to 0.0.
- **W_init** (*initialize function for weight*) – Default to be He truncated normal distribution.
- **b_init** (*initialize function for bias*) – Default to be truncated normal distribution.
- **w_init** (*Optional[Callable[[Sequence[int], Any], jax.numpy.lax_numpy.ndarray]]*) –
- **name** (*Optional[str]*) –

__call__(*node_feats*, *adj*, *is_training*)
Update node features.

Parameters

- **node_feats** (*ndarray of shape (N, in_feats)*) – Batch input node features. N is the total number of nodes in the batch

- **adj** (*ndarray of shape (2, E)*) – Batch adjacency list. E is the total number of edges in the batch
- **is_training** (*bool*) – Whether the model is training or not.

Returns **new_node_feats** – Batch new node features.

Return type ndarray of shape (N, out_feats)

3.2 Readout

pad_graph_pooling(*method='mean'*)

Pooling function for pad pattern graph data.

method [Literal['max', 'min', 'mean', 'sum']] pooling method name.

Returns This function aggregates node_feats about axis=1.

Return type Function

Parameters **method** (*typing_extensions.Literal['max', 'min', 'mean', 'sum']*) –

sparse_graph_pooling(*method='mean'*)

Pooling function for sparse pattern graph data.

method [Literal['max', 'min', 'mean', 'sum']] pooling method name.

Returns This function aggregates node_feats with graph_idx.

Return type Function

Parameters **method** (*typing_extensions.Literal['max', 'min', 'mean', 'sum']*) –

CHAPTER

FOUR

JAXCHEM.UTILS

class EarlyStopping(*patience=10*, *delta=0*, *is_greater_better=True*)

Early stops the training if score doesn't improve after a given patience.

__init__(*patience=10*, *delta=0*, *is_greater_better=True*)

Parameters

- **patience** (*int*) – How long to wait after last time validation loss improved, default to be 10.
- **delta** (*float*) – Minimum change in the monitored quantity to qualify as an improvement, default to be 0.
- **is_greater_better** (*bool*) – Whether the greater score is better or not default to be True.

update(*score*, *checkpoints=None*)

Update early stopping counter.

Parameters

- **score** (*float*) – validation score per epoch.
- **checkpoints** (*Any*) – all parameters and states of training model.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

j

INDEX

Symbols

B

E

J

M

P

S

U