panndas

Charles Frye

Apr 01, 2022

CONTENTS

CHAPTER

ONE

REFERENCE

1.1 panndas.nn

class panndas.nn.**AdditiveSkip**(*block*)

A Module that applies an additive "skip" connection around the provided Module.

forward(*xs*)

Applies the Module to its input.

show()

Displays the Module in a human-friendly format.

class panndas.nn.**AlphaDropout**(*p*, *alpha=0.0*)

A Module that randomly replaces a fraction of its inputs with a fixed value.

(Pseudo-)random values are drawn from the random standard library module.

Parameters

- **p** – The probability that any value is masked on a single call. Sensible values are between 0.0 and 1.0, but this is not checked.
- **alpha** – The value that replaces the masked value. Typically set to the neutral value for a following or preceding non-linearity, e.g. 0.0 for ReLU or sigmoid.

forward(*xs*)

Applies the Module to its input.

show()

Displays the Module in a human-friendly format.

class panndas.nn.**BatchNorm1d**(*eps=1e-05*, *gamma=1.0*, *beta=0.0*)

Standardize each feature across batches and set mean/sd to beta/gamma.

forward(*xs*)

Applies the Module to its input.

class panndas.nn.**Dropout**(*p*)

An AlphaDropout Module with alpha set to 0.0.

See AlphaDropout for details.

show()
Displays the Module in a human-friendly format.

class panndas.nn.**Identity**
A Module that returns its inputs unaltered.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**LayerMaxNorm**
Normalize across the feature dimension with respect to the infinity norm.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**Linear**(*weights_df*, *bias_series=- 1.0*)
A Module that multiplies its inputs by the weights_df and adds the bias_series.

Input 'tensors' can be at most 2-D here: feature (rows) and batch/sequence (columns).

Parameters

- **weights_df** – Weights for the affine transform. Column index is the input feature space and row index is the output feature space.
- **bias_series** – Biases for the affine transform. If not a pd.Series, presumed to be a single element that is promoted to a Series.

Examples

```
>>> import pandas as pd
>>> import panndas.nn as nn
>>> w = pd.DataFrame([[0.0, 1.0],[1.0, 0.0]])              # reflection matrix
>>> w.columns = pd.Index(["left", "right"], name="inputs")
>>> w.index = pd.Index(["right", "left"], name="outputs")  # reflection mirrors␣
↪inputs
>>> l = nn.Linear(weights_df=w, bias_series=0.0)
>>> s = pd.Series([1.0, 2.0], index=w.columns)
>>> s
inputs
left    1.0
right   2.0
dtype: float64
>>> l(s)
outputs
right    2.0
left     1.0
dtype: float64
```

forward(*xs*)
Applies the Module to its input.

show()
Displays the Module in a human-friendly format.

class panndas.nn.**LinearAttention**(*queries_df*, *keys_df*, *values_df*)
The most basic version of an attention layer.

Combines queries, keys, and values linearly.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**Mish**
Applies the Mish function, element-wise.

For details, see Mish: A Self-Regularized Non-Monotonic Neural Activation Function.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**ReLU**
Ol' ReLU-iable.

Applies the recitified linear function, elementwise.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**Sigmoid**
Applies the sigmoid function, element-wise.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**Softmax**
Applies softmax function, column-wise.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**SoftmaxAttention**(*queries_df*, *keys_df*, *values_df*)
The best-known version of an attention layer.

Uses a softmax over the sequence dim to select which values to attend to.

forward(*xs*)
Applies the Module to its input.

class panndas.nn.**Softplus**
Applies the softplus function, element-wise.

forward(*xs*)
Applies the Module to its input.

CHAPTER

TWO

INSTALLATION

```
$ pip install panndas
```

CHAPTER

THREE

USAGE

Simply don’t.

PYTHON MODULE INDEX

INDEX