SimpleNet Documentation

Release v0.1.2

Nathan Henrie

Dec 12, 2017

Contents

Contents:

CHAPTER 1

simplenet package

1.1 Submodules

1.2 simplenet.simplenet module

simplenet.simplenet :: Define SimpleNet class and common functions.

class simplenet.simplenet.**SimpleNet**(*hidden_layer_sizes: typing.Sequence[int], input_shape: typing.Tuple[int, int], output_shape: typing.Tuple[int, int], activation_function: typing.Callable[..., numpy.ndarray] = <function sigmoid>, output_activation: typing.Callable[..., numpy.ndarray] = <function sigmoid>, loss_function: typing.Callable[..., float] = <function neg_log_likelihood>, learning_rate: float = 1.0, dtype: str = 'float32', seed: int = None*) → None

Bases: object

Simple example of a multilayer perceptron.

__init__(*hidden_layer_sizes: typing.Sequence[int], input_shape: typing.Tuple[int, int], output_shape: typing.Tuple[int, int], activation_function: typing.Callable[..., numpy.ndarray] = <function sigmoid>, output_activation: typing.Callable[..., numpy.ndarray] = <function sigmoid>, loss_function: typing.Callable[..., float] = <function neg_log_likelihood>, learning_rate: float = 1.0, dtype: str = 'float32', seed: int = None*) → None

Initialize the MPL.

Parameters

- **hidden_layer_sizes** – Number of neurons in each hidden layer
- **input_shape** – Shape of inputs (m x n), use *None* for unknown m
- **output_shape** – Shape of outputs (m x o), use *None* for unknown m
- **activation_function** – Activation function for all layers prior to output

- **output_activation** – Activation function for output layer
- **learning_rate** – learning rate
- **dtype** – Data type for floats (e.g. np.float32 vs np.float64)
- **seed** – Optional random seed for consistent outputs (for debugging)

export_model (*filename: str*) → None
Export the learned biases and weights to a file.

Saves each weight and bias in order with an index and a prefix of *W* or *b* to ensure it can be restored in the proper order.

Parameters **filename** – Filename for the saved file.

import_model (*filename: str*) → None
Import learned biases and weights from a file.

Parameters **filename** – Name of file from which to import

learn (*inputs: typing.Union[typing.Sequence[int], typing.Sequence[float], numpy.ndarray], targets: typing.Union[typing.Sequence[int], typing.Sequence[float], numpy.ndarray]*) → None
Perform a forward and backward pass, updating weights.

Parameters

- **inputs** – Array of input values
- **targets** – Array of true outputs

predict (*inputs: typing.Union[typing.Sequence[int], typing.Sequence[float], numpy.ndarray]*) → numpy.ndarray
Use existing weights to predict outputs for given inputs.

Note: this method does *not* update weights.

Parameters **inputs** – Array of inputs for which to make predictions

Returns Array of predictions

validate (*inputs: numpy.ndarray*, *targets: numpy.ndarray*, *epsilon: float = 1e-07*) → bool
Use gradient checking to validate backpropagation.

This method uses a naive implementation of gradient checking to try to verify the analytic gradients.

Parameters

- **inputs** – Array of input values
- **targets** – Array of true outputs
- **epsilon** – Small value by which to perturb values for gradient checking

Returns Boolean reflecting whether or not the gradients seem to match

simplenet.simplenet.**cross_entropy** (*y_hat: numpy.ndarray*, *targets: numpy.ndarray*, *der: bool = False*) → float
Calculate the categorical cross entropy loss.

Parameters

- **y_hat** – Array of predicted values from 0 to 1
- **targets** – Array of true values

Returns Mean loss for the sample

simplenet.simplenet.**neg_log_likelihood**(*y_hat: numpy.ndarray*, *targets: numpy.ndarray*, *der: bool = False*) → float

Calculate the negative log likelihood loss.

I believe this is also called the binary cross-entropy loss function.

Parameters

- **y_hat** – Array of predicted values from 0 to 1
- **targets** – Array of true values

Returns Mean loss for the sample

simplenet.simplenet.**relu**(*arr: numpy.ndarray*, *der: bool = False*) → numpy.ndarray

Calculate the relu activation function.

Parameters

- **arr** – Input array
- **der** – Whether to calculate the derivative

Returns Array of outputs from 0 to maximum of the array in a given axis

simplenet.simplenet.**sigmoid**(*arr: numpy.ndarray*, *der: bool = False*) → <built-in function array>

Calculate the sigmoid activation function.

$$\frac{1}{1 + e^{-x}}$$

Derivative:

$$x * (1 - x)$$

Parameters **arr** – Input array of weighted sums

Returns Array of outputs from 0 to 1

simplenet.simplenet.**softmax**(*arr: numpy.ndarray*) → numpy.ndarray

Calculate the softmax activation function.

This equation uses a "stable softmax" that subtracts the maximum from the exponents, but which should not change the results.

$$\frac{e^x}{\sum e^x}$$

Parameters **arr** – Input array of weighted sums

Returns Array of outputs from 0 to 1

1.3 Module contents

simplenet :: Simple multilayer perceptron in Python using numpy.

CHAPTER 2

Credits

2.1 Development Lead

- Nathan Henrie nate@n8henrie.com

2.2 Contributors

- None yet. Why not be the first?

CHAPTER 3

Changelog

3.1 0.1.2 :: 2017-12-12

- Update initialization (now uses something like Xavier)

3.2 0.1.0 :: 2017-11-02

- First release on PyPI / GitHub.

CHAPTER 4

SimpleNet

A simple neural network in Python

- Free software: MIT
- Documentation: https://simplenet-nn.readthedocs.io

4.1 Features

- Simple interface
- Minimal dependencies (numpy)
- Runs on Pythonista on iOS
- Attempts to verify accuracy by comparing results with popular frameworks Keras and Tensorflow

4.2 Introduction

This is a simple multilayer perceptron that I decided to build as I learned a little bit about machine learning and neural networks. It doesn't have many features.

4.3 Dependencies

- Python >= 3.5 (will likely require 3.6 eventually, if Pythonista updates)
- numpy

4.4 Quickstart

1. `pip3 install simplenet`
2. See `examples/`

4.4.1 Development Setup

1. Clone the repo: `git clone https://github.com/n8henrie/simplenet && cd simplenet`
2. Make a virtualenv: `python3 -m venv venv`
3. `source venv/bin/activate`
4. `pip install -e .[dev]`

4.5 Acknowledgements

- Andrew Ng's Coursera courses

4.6 TODO

I don't really know any Latex, so if anybody wants to help me fill out some of the other docstrings with pretty equations, feel free. I'm also not a mathematician, so if anything doesn't seem quite right, feel free to speak up.

4.7 Troubleshooting / FAQ

- How can I install an older / specific version of SimpleNet?
  - Install from a tag:
    - pip install git+git://github.com/n8henrie/simplenet.git@v0.1.0
  - Install from a specific commit:
    - pip install git+git://github.com/n8henrie/simplenet.git@aabc123def456ghi789

CHAPTER 5

Indices and tables

- genindex
- modindex

Python Module Index

S

Index

Symbols

C

E

I

L

N

P

R

S

V