We use cookies on kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using kaggle, you agree to our use of cookies.

Got it Learn more

tfts Oct01-2018 multivariate tensorflow timeseries

Python script using data from Dataset System Resources CPU RAM Disk Network · 208 views

0 Fork 0

Code

This kernel has been released under the Apache 2.0 open source license.

```
'''
Dated: Oct01-2018
Author: Mahesh Babu Mariappan (https://www.linkedin.com/in/mahesh-babu-mariappan)
Source code for simultaneously forecasting multiple time series taking into account their corre
Dataset used: systemresources-deeplearning-1000.csv
Columns:
index   (timesteps)
cpu     (utilization)
ram     (utilization)
disk        (utilization)
network     (utilization)


I am bounding the forecasts between 0 and 100 (because we are looking at forecasting resource u

Warning: program may consume a lot of cpu and ram.

Results:
If you have matplotlib installed, you should see a visualization of 1000 past timesteps, and 10
Time elapsed on an intel core i5 4-core cpu, 8gb ram: 00h:01m:18.26s


'''
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import matplotlib
from matplotlib import pyplot as plt
import numpy
import time
from os import path
import tempfile
import tensorflow as tf


_PATH = path.dirname(__file__)
_CSV_FILE = path.join(_PATH, '../input/systemresources-deeplearning-1000.csv')

def bound_forecasts_between_0_and_100(ndarray):
  '''I am bounding the forecasts between 0 and 100 (because we are looking at forecasting resou
  return numpy.clip(ndarray, 0, 100)

def multiple_timeseries_forecast(
    csv_file_name=_CSV_FILE, export_directory=None, training_steps=500):
  '''Trains and evaluates a tensorflow model for simultaneously forecasting multiple time serie
  estimator = tf.contrib.timeseries.StructuralEnsembleRegressor(
      periodicities=[], num_features=4)
  reader = tf.contrib.timeseries.CSVReader(
      csv_file_name,
      skip_header_lines=1,
      column_names=((tf.contrib.timeseries.TrainEvalFeatures.TIMES,)
                    + (tf.contrib.timeseries.TrainEvalFeatures.VALUES,) * 4))
  train_input_fn = tf.contrib.timeseries.RandomWindowInputFn(
      reader, batch_size=4, window_size=64)
  estimator.train(input_fn=train_input_fn, steps=training_steps)
  evaluation_input_fn = tf.contrib.timeseries.WholeDatasetInputFn(reader)
  current_state = estimator.evaluate(input_fn=evaluation_input_fn, steps=1)
  values = [current_state["observed"]]
  times = [current_state[tf.contrib.timeseries.FilteringResults.TIMES]]
  if export_directory is None:
    export_directory = tempfile.mkdtemp()
  input_receiver_fn = estimator.build_raw_serving_input_receiver_fn()
  export_location = estimator.export_savedmodel(
      export_directory, input_receiver_fn)
  with tf.Graph().as_default():
    numpy.random.seed(1)
    with tf.Session() as session:
      signatures = tf.saved_model.loader.load(
          session, [tf.saved_model.tag_constants.SERVING], export_location)
      for _ in range(100):
```