Machine Learning Applications

Time Series Prediction with Convolutional Neural Networks and Keras

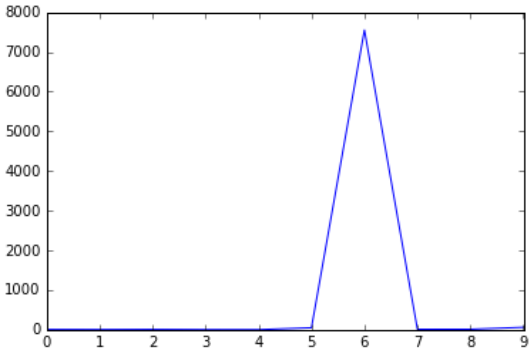
June 17, 2017 by owygs156

A convolutional neural network (CNNs) is a type of network that has recently

gained popularity due to its success in classification problems (e.g. image recognition
or time series classification) [1]. One of the working examples how to use Keras CNN for time series can be found at this link[2]. This example allows to predict for single time series and multivariate time series.

Running the code from this link, it was noticed that sometimes the prediction error has very high value, may be because optimizer gets stuck in local minimum. (See Fig1. The error is on axis Y and is very high for run 6) So I updated the script to run several times and then remove results with high error. (See Fig2 The Y axis showing small error values). Here is the summary of all changes:

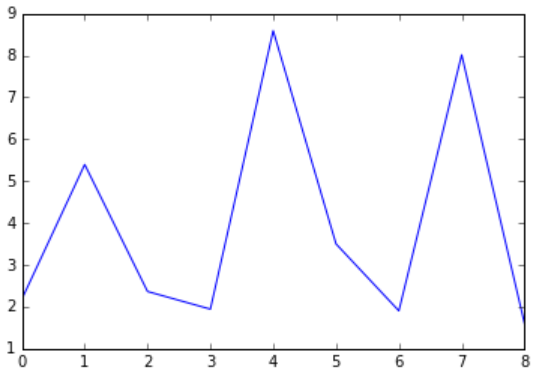
- Created multiple runs that can allow to filter bad results based on error. Training CNN is running 10 times and for each run error data and some other associated data is saved. Error is calculated as square root of sum if squared errors for last 10 predictions during the training.
- Added also plot to see error over multiple runs.
- In the end of script added one plot that showing errors for each run (See Fig1.) , and another plot showing errors only for runs that did not have high error (See Fig2.).
- Added saving keras model to file for each run and then loading it from file for model that showed best results (min error). See [3] for more information on saving and loading keras models.
- Added ability to load time series from csv file.



Fun Machine Learning Projects and Discussions with a Purpose[NEW] **- How to do ChatBots,**

Error for all runs

Fig 1.



Error chart after removing runs with high value error

Fig 2.

Below is the full code

```
#!/usr/bin/env python
"""
This code is based on convolutional neural network
model from below link
gist.github.com/jkleint/1d878d0401b28b281eb75016ed2
9f2ee
"""

from __future__ import print_function, division

import numpy as np
from keras.layers import Convolution1D, Dense, MaxP
ooling1D, Flatten
from keras.models import Sequential
from keras.models import model_from_json

import matplotlib.pyplot as plt
import csv
```


Recent Posts

```
__date__ = '2017-06-22'

error_total =[]
result=[]
i=0

def make_timeseries_regressor(window_size, filter_l
ength, nb_input_series=1, nb_outputs=1, nb_filter=4
):
    """:Return: a Keras Model for predicting the next

    The model can handle multiple input timeseries (`n

    :param int window_size: The number of previous tim
    :param int nb_input_series: The number of input ti
      The `X` input to ``fit()`` should be an array of
      a 2D array of shape ``(window_size, nb_input_ser
      single timeseries), one instance could be ``[[0]
    :param int nb_outputs: The output dimension, often
      For each input instance (array with shape ``(win
      usually the value(s) predicted to come after the
      in the sequence. The `y` input to ``fit()`` shou
    :param int filter_length: the size (along the `win
      each position along each instance. The differenc
      to the number of input timeseries (its "width" b
      dimension.  This is useful as generally the inpu
      meaningful to look for patterns that are invaria
    :param int nb_filter: The number of different filt
    """
    model = Sequential((
        # The first conv layer learns `nb_filter` filt
        # Its output will have shape (None, window_siz
        # the input timeseries, the activation of each
        Convolution1D(nb_filter=nb_filter, filter_leng
        MaxPooling1D(),     # Downsample the output of
        Convolution1D(nb_filter=nb_filter, filter_leng
        MaxPooling1D(),
        Flatten(),
        Dense(nb_outputs, activation='linear'),     #
    ))
    model.compile(loss='mse', optimizer='adam', metric
```

Learning Python DQN Application for Resource Allocation

Application of Daubechies Wavelet for Denoising 1D Data

Integrating Sentiment Analysis API Python Django into Web Application

```
    # To perform (binary) classification instead:
    # model.compile(loss='binary_crossentropy', optimi
    return model


def make_timeseries_instances(timeseries, window_si
ze):
    """Make input features and prediction targets from

    :return: A tuple of `(X, y, q)`.  `X` are the inpu
      ``(timeseries.shape[0] - window_size, window_siz
      corresponding row of `y` is the next value in th
      to predict a hypothetical next (unprovided) valu
    :param ndarray timeseries: Either a simple vector,
      row) and the series is axis 1 (the column).
    :param int window_size: The number of samples to u
    """
    timeseries = np.asarray(timeseries)
    assert 0 < window_size < timeseries.shape[0]
    X = np.atleast_3d(np.array([timeseries[start:start
    y = timeseries[window_size:]
    q = np.atleast_3d([timeseries[-window_size:]])
    return X, y, q


def evaluate_timeseries(timeseries, window_size):
    """Create a 1D CNN regressor to predict the next v
    as input features and evaluate its performance.

    :param ndarray timeseries: Timeseries data with ti
    :param int window_size: The number of previous tim
    """
    filter_length = 5
    nb_filter = 4
    timeseries = np.atleast_2d(timeseries)
    if timeseries.shape[0] == 1:
        timeseries = timeseries.T       # Convert 1D v

    nb_samples, nb_series = timeseries.shape
    print('\n\nTimeseries ({} samples by {} series)
:\n'.format(nb_samples, nb_series), timeseries)
```


Top Posts & Pages

```
    model = make_timeseries_regressor(window_size=wind
    print('\n\nModel with input size {}, output siz
e {}, {} conv filters of length {}'.format(model.in
put_shape, model.output_shape, nb_filter, filter_le
ngth))
    model.summary()

    error=[]

    X, y, q = make_timeseries_instances(timeseries, wi
    print('\n\nInput features:', X, '\n\nOutput lab
els:', y, '\n\nQuery vector:', q, sep='\n')
    test_size = int(0.01 * nb_samples)           # In
    X_train, X_test, y_train, y_test = X[:-test_size],
    model.fit(X_train, y_train, nb_epoch=25, batch_siz

    # serialize model to JSON
    model_json = model.to_json()
    with open("model"+str(i)+".json", "w") as json_fil
          json_file.write(model_json)
    # serialize weights to HDF5
    model.save_weights("model"+str(i)+".h5")
    print("Saved model to disk")
    global i
    i=i+1

    pred = model.predict(X_test)
    print('\n\nactual', 'predicted', sep='\t')
    error_curr=0
    for actual, predicted in zip(y_test, pred.squeeze(
        print(actual.squeeze(), predicted, sep='\t'
)
        tmp = actual-predicted
        sum_squared = np.dot(tmp.T , tmp)
        error.append ( np.sqrt(sum_squared) )
        error_curr=error_curr+ np.sqrt(sum_squared)
    print('next', model.predict(q).squeeze(), sep='
\t')
    result.append  (model.predict(q).squeeze())
    error_total.append (error_curr)
    print (error)
```

Mining

Building Decision Trees in Python - Handling Categorical Data

Applied Machine Learning Classification for Decision Making

LSTM Neural Network Training - Few Useful Techniques for Tuning Hyperparameters and Saving Time

Machine Learning Stock Prediction with LSTM and Keras

Retrieving Emails from POP3 Server Using Python Script

3 Most Useful Examples to Add Interactivity to Graph Data

```
def read_file(fn):
    '''
    Reads the CSV file
    -----
    RETURNS:
        A matrix with the file contents
    '''

    vals = []
    with open(fn, 'r') as csvfile:
        tsdata = csv.reader(csvfile, delimiter=',')
        for row in tsdata:
             vals.append(row)

    # removing title row
    vals = vals[1:]
    y = np.array(vals).astype(np.float)
    return y
```

```
def main():
    """Prepare input data, build model, eval uate."""
    np.set_printoptions(threshold=25)
    ts_length = 1000
    window_size = 50
    number_of_runs=10
    error_max=200

    print('\nSimple single timeseries vector predic
tion')
    timeseries = np.arange(ts_length)
    # enable below line to run this time series
    #evaluate_timeseries(timeseries, window_size)

    print('\nMultiple-input, multiple-output predic
```

Using Bokeh Library

Time Series Prediction with LSTM and Keras for Multiple Steps Ahead

Random Forest Classifier with Python

Neural Networks with Python on the Web

Recently added many new links about neural nets for encoder - decoder, text translation, summarization , question answering , image recognition and other applications.

```
tion')
    timeseries = np.array([np.arange(ts_length), -np.a
    # enable below line to run this time series
    ##evaluate_timeseries(timeseries, window_size)

    print('\nMultiple-input, multiple-output predic
tion')
    timeseries = np.array([np.arange(ts_length), -np.a
    # enable below line to run this time series
    #evaluate_timeseries(timeseries, window_size)

    timeseries = read_file('ts_input.csv')
    print (timeseries)

    for i in range(number_of_runs):
        evaluate_timeseries(timeseries, window_size)

    error_total_new=[]
    for i in range(number_of_runs):
        if (error_total[i] < error_max):
            error_total_new.append (error_total[i])

    plt.plot(error_total)
    plt.show()
    print (result)

    plt.plot(error_total_new)
    plt.show()
    print (result)

    best_model=np.asarray(error_total).argmin(axis=0)
    print ("best_model="+str(best_model))




    json_file = open('model'+str(best_model)+'.json',
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
```

Archives

```
        # load weights into new model
        loaded_model.load_weights("model"+str(best_model)+
        print("Loaded model from disk")



if __name__ == '__main__':
    main()
```

References

1. Conditional Time Series Forecasting with Convolutional Neural Networks
2. Example of using Keras to implement a 1D convolutional neural network (CNN) for timeseries prediction
3. Save and Load Your Keras Deep Learning Models

Print Models Activations Patterns Return Images Archives C
Allows Applications Print Models Activations Patterns Retu

Share this:

Related

Forecasting Time Series Data with Convolutional Neural Networks
May 14, 2017
In "Artificial Intelligence"

Time Series Prediction with LSTM and Keras for Multiple Steps Ahead
February 27, 2018
In "Machine Learning"

Iris Plant Classification Using Neural Network - Online Experiments with Normalization and Other Parameters
January 28, 2017
In "Artificial Intelligence"

April 2017 (2)
March 2017 (2)
February 2017 (5)
January 2017 (4)
December 2016 (4)
November 2016 (1)
October 2016 (3)
September 2016 (3)
August 2016 (5)
July 2016 (3)
June 2016 (2)
May 2016 (3)
April 2016 (1)
March 2016 (2)
February 2016 (18)
January 2016 (5)

Artificial Intelligence, Machine Learning, Python Scripts

convolutional neural networks, deep learning, machine learning, time series, time series prediction

‹ Forecasting Time Series Data with Convolutional Neural Networks

› Algorithms, Metrics and Online Tool for Clustering

Leave a Comment

You must be logged in to post a comment.