keras-team / keras

Input 0 is incompatible with layer lstm_1: expected ndim=3, found ndim=4 #7403

New issue

Open ajanaliz opened this issue on Jul 22, 2017 · 17 comments

ajanaliz commented on Jul 22, 2017

Here's the code I've written:

```
model.add(LSTM(150,
               input_shape=(64, 7, 339),
               return_sequences=False))
model.add(Dropout(0.2))

model.add(LSTM(
    200,
    return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(
    150,
    return_sequences=True))
model.add(Dropout(0.2))

model.add(Dense(
    output_dim=1))
model.add(Activation('sigmoid'))

start = time.time()
model.compile(loss='mse', optimizer='rmsprop')
print('compilation time : ', time.time() - start)

model.fit(
    trainX,
    trainY_Buy,
    batch_size=64,
    nb_epoch=10,
    verbose=1,
    validation_split=0.05)
```

the error i'm getting is this:
ValueError: Input 0 is incompatible with layer lstm_1: expected ndim=3, found ndim=4
on this line:
model.add(LSTM(150,
input_shape=(64, 7, 339),
return_sequences=False))

my X shape is: (492, 7, 339)
my Y shape is: (492,)

anyone have any ideas on what I'm doing wrong?

Assignees

No one assigned

Labels

stale

Projects

None yet

Milestone

No milestone

Notifications

7 participants

ajanaliz commented on Jul 22, 2017 • edited

same thing happens when I wrote the following for the first LSTM layer:

```
model.add(LSTM(150,
               input_shape=trainX.shape,
               return_sequences=False))
```

td2014 commented on Jul 22, 2017 Contributor

@ajanaliz . I took a quick look, and I believe that you need to remove the leading "64" from the input shape of the LSTM layer --> input_shape=(64, 7, 339), --> input_shape=(7, 339).
Keras' convention is that the batch dimension (number of examples (not the same as timesteps)) is typically omitted in the input_shape arguments. The batching (number of examples per batch) is handled in the fit call. I hope that helps. Thanks.

4

ajanaliz commented on Jul 22, 2017

@td2014 nope, that way my error is:
Input 0 is incompatible with layer lstm_2: expected ndim=3, found ndim=2

td2014 commented on Jul 22, 2017 Contributor

@ajanaliz . You may need to turn "return_sequences=True" in the first layer. Maybe that will solve it. I hope that works. Thanks.

10

ajanaliz commented on Jul 22, 2017

omg, thanks

StephanCHEN commented on Aug 2, 2017

Hi, I have the same bug as you. My X shape is (24443, 124, 30), y shape is (24443, 124). May be it's the shape of y that causes the error for me. May I know the type of your Y?

ajanaliz commented on Aug 3, 2017

ur y shape should be (24443,)

Sent from myMail for iOS

Wednesday, August 2, 2017, 6:08 PM +0430 from notifications@github.com <notifications@github.com>:
...

StephanCHEN commented on Aug 3, 2017

I get an error even when I construct the same keras layer as yours...
"Errors when checking target: expected activation_1 to have 3 dimensions, but got array with shape (24443, 1)"

ajanaliz commented on Aug 4, 2017

i think it has to do with ur lstm architecture and the way you're returning feedbacks

Sent from myMail for iOS

Thursday, August 3, 2017, 11:34 AM +0430 from notifications@github.com <notifications@github.com>:
...

atishsawant commented on Sep 1, 2017

Having the same issue with the following code. Was there ever a fix for this?

```
model.add(LSTM(32, input_shape=(588425, 26), return_sequences = True))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(df_matrix, y, epochs=2, batch_size=1, verbose=2)
```

kicking off this error: ValueError: Input 0 is incompatible with layer lstm_23: expected ndim=3, found ndim=2

These are my inputs: df_matrix.shape = (1, 588425, 26)

I tried it as a 2-D arry as well and I get the same error. What am I doing wrong here? Running on Windows 10, latest Keras/TF backend

hadisaadat commented on Sep 3, 2017

Dear **@td2014** could you give me a hand
Actually the real problem is this I want to have 2 stacked lstm layers in the way that :

- first layer read directly the input data as a sequence of characters a 3D tensor with shape: (batch_size, timesteps=Max_word_length, input_dim=Char_embedding_size) and its output(word) at the end of each sequence(length of word) is a 2D tensor with shape (batch_size, output_dim=Max_word_length) and (return_sequences=False) which goes directly to the second lstm.
- Second layer recieves the words sequences from previous layer a 3D tensor with shape: (batch_size/Max_word_length, timesteps=Max_sentence_length, input_dim=Char_embedding_size*Word–Max_length) and it outputs for each word a vector of 300 real number (which is in fact its word embedding vector) so the output is a 2D tensor with shape (batch_size/Max_word_length, output_dim=62) and here (return_sequences=True) ; or I should not shape the input for second layer and keras manage it itself?
- And what should be the final layer ? is it a regression or ...?

its my understanding as sample sudo code but could you please guide me a little more what are wrong in it?

```
batch_size=900
Max_word_len=30
Max_sentence_length=#some custom number
Char_embedding_Size=len(chars)
print('Build model...')
model = Sequential()
#return_sequences=False to output at the end of each sequence
model.add(LSTM(batch_size, input_shape=(Max_word_len, Char_embedding_Size), return_sequences=False))
model.add(LSTM(batch_size/Max_word_len, input_shape=(Max_sentence_length, Max_word_len),
return_sequences=True))
model.add(Dense(62))
model.add(Activation('tanh'))
```

Any hint suggestion and implementation in keras and tensorflow is welcome Thanks a lot again in advance.

Best regards,

Hadi.

td2014 commented on Sep 3, 2017 • edited ▾ Contributor

@hadisaadat . Some suggestions that might help provide some direction for you. I would suggest breaking up your problem into several pieces to make sure you understanding the input/output dimensions of each layer (reviewing this section will help https://keras.io/layers/recurrent/). Also, you have "batch_size" as the first argument of the LSTM call:

model.add(LSTM(batch_size, input_shape=(Max_word_len, Char_embedding_Size), return_sequences=False))

The first argument (which is called "units" in the documentation) is the output dimensionality of that layer. In order words, you will have "units" number of LSTM cells at that layer. In Keras, the batch dimension is not typically specified in the model architecture.

It might be useful to just have a single LSTM layer that feeds into a single Dense layer (with number of units=1, so Dense(1), not Dense(62)) and look at the architecture you get from model.summary()).

Also, the "return_sequences=True" for each sample, will generate output from each LSTM cell for each timestep. This is typically fed into a second RNN layer, not into a regular Dense layer. Also, Keras should automatically infer the input data shape for every layer except the first.

If you are trying to processing characters -> words, and then words --> sentences, it might be easier to create two separate models (each one with a single LSTM layer) instead of directly trying to stack, because your batch definition is changing between layers.

Also, as far as the final layer goes, that will depend on what your target is? In other words, what do you want your model to output. Are you trying to classify the type of sentence? From your above architecture, it seems you have 62 different outputs/classes you are trying to model. Depending on what your target is, this should help you decide on the final output layer.

I hope this gives you a few ideas to help. Thanks.

 1

hadisaadat commented on Sep 4, 2017 • edited ▾

Dear **@td2014** thank you so much for your reply,

Actually yes Im doing some hierarchical stacked layers for text as you mentioned Chars->word and then word->vec[0..61] (a vector of real number)

> If you are trying to processing characters -> words, and then words --> sentences

yes your right it is easier but I want to have this hierarchical structure to see even its effect in comparison with having 2 separate model as you suggested

> it might be easier to create two separate models (each one with a single LSTM layer) instead of directly trying to stack

is it a big challenge to fit such an issue? I mean isn't it possible to have 2 stacked layer with relative batch_size by some option like None keyword

> because your batch definition is changing between layers

specially here there are some unclear points let say I have such a sudo code

- data_dim = len(chars)
- timesteps = 40 #the maximum length(characters) of a word
- out_vec_size = 62
- num_hidden_neurons_char_layer = timesteps
- num_hidden_neurons_word_layer=out_vec_size
- batch_size=800
- model = Sequential()

- model.add(LSTM(num_hidden_neurons_char_layer, return_sequences=False, batch_input_shape=(None, timesteps, data_dim)))
- model.add(LSTM(num_hidden_neurons_word_layer))
- model.add(Dense(out_vec_size, activation='tanh'))
-

It's my 3 layers input and output shapes of the above model:
(None, 40, 57) (None, 40, 40)
(None, 40, 40) (None, 62)
(None, 62) (None, 62)

in the first lstm layer at charachter level I want the layer output only at the end of sequence which is the end on the word(let assume I have a padded) so I set return_sequences=False to force it not to output for each input character only at the end, on the other hand I want the second layer receive input for each word and out put for each word as well, so return_sequences=True is for this layer; some questions raises here :

1. what is really relation between the batch_size here with return_sequences=False/True ?

2. if dont force the first layer to output at the end of seq it works but it's not what I need and when I force it, I receive the dimension error of second layer input ,how to solve it?

and my final out put is just a real number vector for each word it's not a class so I do not need a softmax like layer at the end to calculate the probability for each class, its like a regression to predict a real number but with dimension 62.

> Also, as far as the final layer goes, that will depend on what your target is? In other words, what do you want your model to output. Are you trying to classify the type of sentence? From your above architecture, it seems you have 62 different outputs/classes you are trying to model. Depending on what your target is, this should help you decide on the final output layer.

However if its not possible with the keras architecture any suggestion in tensorflow is welcome.
Finally I want to appreciate again for your attention and explanations.

Hadi.

td2014 commented on Sep 5, 2017 • edited ▾ Contributor

@hadisaadat . Here is a bit more information which might help, based on your info above:

> in the first lstm layer at charachter level I want the layer output only at the end of sequence which is the end on the word(let assume I have a padded) so I set return_sequences=False to force it not to output for each input character only at the end, on the other hand I want the second layer receive input for each word and out put for each word as well, so return_sequences=True is for this layer; some questions raises here :

> what is really relation between the batch_size here with return_sequences=False/True ?

The basic layout is the following:

Batch_sample_1: timestep1, timestep2, ..., timestepN
Batch_sample_2: timestep1, timestep2, ..., timestepN
...
Batch_sample_BatchSize: timestep1, timestep2, ..., timestepN

When you set return_sequences=True, then for each batch_sample, you get outputs for each time step (And this will be one output for each LSTM cell in that layer).
When you set return_sequences=False, then for each batch_sample, you get the output for timestepN only.
In the second layer, since you are doing input and output for each word (which is from layer1), return_sequences=False is what you want I think (you only want the output from each input word, not the character that forms the word. For each sequence of timestep1...timestepN, you only have one word predicted).

Some more ideas based on using Masking and Merge Layers that might suggest some direction:

```
from keras.layers import LSTM, Input, Masking, multiply
from keras.models import Model

#
# Create input sequences
#
numTimesteps=20
slopeArray1=np.linspace(0, 10, num=numTimesteps)
slopeArray1 = np.expand_dims(slopeArray1, axis=0)
slopeArray1 = np.expand_dims(slopeArray1, axis=2)

slopeArray2=np.linspace(0, 15, num=numTimesteps)
slopeArray2 = np.expand_dims(slopeArray2, axis=0)
slopeArray2 = np.expand_dims(slopeArray2, axis=2)
maskArray=np.zeros((1,numTimesteps,1))
maskArray[0,numTimesteps-1]=1

X_train = np.concatenate((slopeArray1, slopeArray2))
X_mask = np.concatenate((maskArray, maskArray))

# preparing y_train
y_train = []
y_train = np.array([2*slopeArray1[0,19]-slopeArray1[0,18],
                    2*slopeArray2[0,19]-slopeArray2[0,18]]) # make target one delta higher

#
# Create model
#

inputs = Input(name='Input1', batch_shape=(1,numTimesteps,1))
X_mask_input = Input(name='Input2', batch_shape=(1,numTimesteps,1))
x = LSTM(units=1, name='LSTM1', return_sequences=True)(inputs)
x = multiply([x, X_mask_input])
x = Masking(mask_value=0.0)(x)
pred = LSTM(units=1, name='LSTM2', return_sequences=False, stateful=True)(x)
model = Model(inputs=[inputs, X_mask_input], outputs=pred)
model.compile(loss='mse', optimizer='sgd', metrics=['mse'])
print(model.summary())

#
# Train
#
model.fit([X_train, X_mask], y_train, epochs=200, batch_size=1)
```

The idea is that if you can set a value (say 0.0) to be an "ignore" using the mask, then the second LSTM will only process on the final output, I think.
More details here:https://keras.io/layers/core/

I hope this helps. Thanks.

stale bot commented on Dec 5, 2017

This issue has been automatically marked as stale because it has not had recent activity. It will be closed after 30 days if no further activity occurs, but feel free to re-open a closed issue if needed.

stale bot added the **stale** label on Dec 5, 2017

karan10111 commented 21 days ago

model.add(LSTM(output_dim,
input_shape=(tsteps, 1),
batch_size=batch_size,
return_sequences=True,
stateful=True))

Giving the batch size explicitly solved the same for me.

rajasrajeev commented 9 days ago • edited

Input 0 is incompatible with layer conv1d_14: expected ndim=3, found ndim=2

please help me with this