Validation loss and accuracy remains constant

Ask Question

8

I am trying to implement this paper http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0137036 on a set of medical images. I am doing it in Keras. The network essentially consists of 4 conv and max-pool layers, followed by a fully connected layer, followed by a soft max classifier. As far as I can see, I have followed the architecture mentioned in the paper. However, the validation loss and accuracy just remains flat throughout. The accuracy seems to be fixed at ~57.5%. Any help on where I could be going wrong would be

```
from keras.layers import Convolution2D, MaxPooling2D
from keras.optimizers import SGD
from keras.utils import np_utils
from PIL import Image
import numpy as np
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
import theano
import os
import glob as glob
import cv2
from matplotlib import pyplot as plt

nb_classes = 2
img_rows, img_cols = 100,100
img_channels = 3


#################### DATA DIRECTORY SETTING#####################

data = '/home/raghuram/Desktop/data'
os.chdir(data)
file_list = os.listdir(data)
#################################################################

## Test lines
#I = cv2.imread(file_list[1000])
#print np.shape(I)
####
non_responder_file_list = glob.glob('0_*FLAIR_*.png')
responder_file_list = glob.glob('1_*FLAIR_*.png')
print len(non_responder_file_list),len(responder_file_list)

labels = np.ones((len(file_list)),dtype = int)
labels[0:len(non_responder_file_list)] = 0
immatrix = np.array([np.array(cv2.imread(data+'/'+image)).flatten() for image in
file_list])
#img = immatrix[1000].reshape(100,100,3)
#plt.imshow(img,cmap = 'gray')


data,Label = shuffle(immatrix,labels, random_state=2)
train_data = [data,Label]
X,y = (train_data[0],train_data[1])
# Also need to look at how to preserve spatial extent in the conv network
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=4)
X_train = X_train.reshape(X_train.shape[0], 3, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 3, img_rows, img_cols)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()

## First conv layer and its activation followed by the max-pool layer#
model.add(Convolution2D(16,5,5, border_mode = 'valid', subsample = (1,1), init =
'glorot_normal',input_shape = (3,100,100))) # Glorot normal is similar to Xavier
initialization
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2),strides = None))
# Output is 48x48

print 'First layer setup'
############################Second conv layer#################################
```

```
model.add(Convolution2D(32,3,3,border_mode = 'same', subsample = (1,1),init =
'glorot_normal'))
model.add(Activation('relu'))
model.add(Dropout(0.6))
model.add(MaxPooling2D(pool_size = (2,2),strides = None))
#############################################################################

print ' Second layer setup'
# Output is 2x24

##########################Third conv layer###################################
model.add(Convolution2D(64,3,3, border_mode = 'same', subsample = (1,1), init =
'glorot_normal'))
model.add(Activation('relu'))
model.add(Dropout(0.6))
model.add(MaxPooling2D(pool_size = (2,2),strides = None))
#############################################################################
# Output is 12x12

print ' Third layer setup'
##############################Fourth conv layer##############################
model.add(Convolution2D(128,3,3, border_mode = 'same', subsample = (1,1), init =
'glorot_normal'))
model.add(Activation('relu'))
model.add(Dropout(0.6))
model.add(MaxPooling2D(pool_size = (2,2),strides = None))
#############################################################################

print 'Fourth layer setup'

# Output is 6x6x128
# Create the FC layer of size 128x6x6#
model.add(Flatten())
model.add(Dense(2,init = 'glorot_normal',input_dim = 128*6*6))
model.add(Dropout(0.6))
model.add(Activation('softmax'))

print 'Setting up fully connected layer'
print 'Now compiling the network'
sgd = SGD(lr=0.01, decay=1e-4, momentum=0.6, nesterov=True)
model.compile(loss = 'mse',optimizer = 'sgd', metrics=['accuracy'])

# Fit the network to the data#
print 'Network setup successfully. Now fitting the network to the data'
model. fit(X_train,Y_train,batch_size = 100, nb_epoch = 20, validation_split =
None,verbose = 1)
print 'Testing'
loss,accuracy = model.evaluate(X_test,Y_test,batch_size = 32,verbose = 1)
print "Test fraction correct (Accuracy) = {:.2f}".format(accuracy)
```

machine-learning python deep-learning keras

edited Aug 24 '16 at 5:52

asked Aug 23 '16 at 6:19

Raghuram

351 ● 2 ● 6 ● 20

Is the training loss going down? – Jan van der Vegt Aug 23 '16 at 7:41

No, the training loss also remains constant throughout. – Raghuram Aug 23 '16 at 13:21

You haven't set any validation data or validation_split in your fit call, what would it validate on? Or did you mean test? – Jan van der Vegt Aug 23 '16 at 13:24

That is after experimenting around. I set validation_split = 0.2 before setting it to None and experimented around with that also. – Raghuram Aug 23 '16 at 15:30

2 Can you fit one batch for a lot of times to see if you can get the training loss to be lower? – Jan van der Vegt Aug 23 '16 at 16:18

2 Answers

3 It seems that you use MSE as the loss function, from a glimpse on the paper it seems they use NLL (cross entropy), MSE is considered prone to be sensitive to data imbalance among other issues and it may be the cause of the problem you experience, I would try training using categorical_crossentropy loss in your case, moreover learning rate of 0.01 seems too large I would try to play with it and try 0.001 or even 0.0001

answered Feb 21 '18 at 21:41

koltun
76 • 2

1

Though I am a bit late here, I would like to put my two cents in as it helped me solve a similar issue recently. What came to my rescue was scaling the features into (0,1) range besides the categorical cross-entropy loss. Nevertheless, it is worth saying that feature scaling helps only if the features belong to different metrics and possess way more variation (in orders of magnitudes) relative to one another, as was in my case. Also, scaling could be really useful if one uses the `hinge` loss, as max-margin classifiers are generally sensitive to the distances among feature values. Hope this helps some future visitors!

answered Jul 19 '18 at 4:39

Saurav--
131 • 3