

Project Notes

Samuel Navarro

June 21, 2019

Contents

1 Visualizing Loss	1
2 Generators	1

1 Visualizing Loss

When calling `model.fit()` or `model.fit_generator()`, Keras outputs a history object that contains the training and validation loss for each epoch.

Code example in Listing 1:

2 Generators

Generators can be a great way to work with large amounts of data. Instead of storing the preprocessed data in memory all at once, using a generator you can pull pieces of the data and process them on the fly only when you need them, which is much more memory-efficient.

```

1 from keras.models import Model
2 import matplotlib.pyplot as plt
3
4 history_object = model.fit_generator(train_generator, samples_per_epoch =
5     len(train_samples), validation_data =
6     validation_generator,
7     nb_val_samples = len(validation_samples),
8     nb_epoch=5, verbose=1)
9
10 ### print the keys contained in the history object
11 print(history_object.history.keys())
12
13 ### plot the training and validation loss for each epoch
14 plt.plot(history_object.history['loss'])
15 plt.plot(history_object.history['val_loss'])
16 plt.title('model mean squared error loss')
17 plt.ylabel('mean squared error loss')
18 plt.xlabel('epoch')
19 plt.legend(['training set', 'validation set'], loc='upper right')
20 plt.show()

```

Listing 1: Visualizing Loss

```

1 def fibonacci()
2     numbers_list = []
3     while 1:
4         if(len(numbers_list) < 2):
5             numbers_list.append(1)
6         else:
7             numbers_list.append(numbers_list[-1] + numbers_list[-2])
8         yield numbers_list # change this line so it yields its list instead of 1
9
10 our_generator = fibonacci()
11 my_output = []
12
13 for i in range(10):
14     my_output = (next(our_generator))
15
16 print(my_output)

```

Listing 2: Generators