How to set class weights for imbalanced classes in Keras?

I know that there is a possibility in Keras with the `class_weights` parameter dictionary at fitting, but I couldn't find any example. Would somebody so kind to provide one?

By the way, in this case the appropriate praxis is simply to weight up the minority class proportionally to its underrepresentation?

classification keras weighted-data

asked Aug 17 '16 at 9:35
Hendrik
614 1 8 20

Is there a new updated method out using Keras ? why is the dictionary consisting of three classes and for class: 0: 1.0 1: 50.0 2: 2.0 ???? shouldn't: 2:1.0 as well ? – Chuck Sep 9 at 14:00

4 Answers

If you are talking about the regular case, where your network produces only one output, then your assumption is correct. In order to force your algorithm to treat every instance of **class 1** as 50 instances of **class 0** you have to:

1. Define a dictionary with your labels and their associated weights

```
class_weight = {0 : 1.,
    1: 50.,
    2: 2.}
```

2. Feed the dictionary as a parameter:

```
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32, class_weight = class_weight)
```

edited Jul 11 at 11:15
Desire
151 2 3 12

answered Aug 17 '16 at 10:49
layser
376 3 2

Also have a look at github.com/fchollet/keras/issues/3653 if you're working with 3D data. – herve Apr 26 at 9:12

For me it gives a error dic don't has shape attribute. – Flávio Filho May 23 at 0:11

I believe Keras could be changing the way this works, this is for the version of August 2016. I will verify for you in a week – layser May 25 at 14:12

You could simply implement the class_weight from sklearn:

1. Let's import the module first

```
from sklearn.utils import class_weight
```

2. In order to calculate the class weight do the following

```
class_weight = class_weight.compute_class_weight('balanced',
np.unique(y_train), y_train)
```

3. Thirdly and lastly add it to the model fitting

```
model.fit(X_train, y_train, class_weight=class_weight)
```