Ex 02 - Keras Confusion Matrix / Validation Set

We'll print the confusion matrix and accuracy.

Lots of the new code is boiler plate to format the confusion matrix.

To keep training fast, we cut off the validation set to a small number of the 500 samples we read.
Presently we are not going through the validation set in batches, just pass all to model.predict() function

model.predict gives the softmax results - looks like a probability distribution over labels 0,1

Code

ex02_keras_train.py

Run

This model should train pretty fast

Discuss

What kind of confusion matrix would be ideal, given that not all 'lasing' images lased?

Can we trust model? Data comes from different runs?

Are you seeing big swings in the confusion matrix?

Big swings in the accuracy? What might help?

Explore Model

A fun way to explore the model is to embed IPython at some point. Let's do the following, and then run on an interactive node (not on batch)

stop in IPython at some step in training:

```
    for epoch in range(3):
        ex01.shuffle_data(training_X, training_Y)
        next_sample_idx = -minibatch_size
        for batch in range(batches_per_epoch):
            if step==4:
                import IPython
                IPython.embed()
            step += 1
```

Setup up Plotting

execute:
%pylab

Look at model:

model.summary()

output should be:

```
In [2]: model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 2, 363, 284)	32	convolution2d_input_1[0][0]
batchnormalization_1 (BatchNormal	(None, 2, 363, 284)	4	convolution2d_1[0][0]
activation_1 (Activation)	(None, 2, 363, 284)	0	batchnormalization_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 2, 90, 71)	0	activation_1[0][0]
convolution2d_2 (Convolution2D)	(None, 6, 90, 71)	192	maxpooling2d_1[0][0]
batchnormalization_2 (BatchNormal	(None, 6, 90, 71)	12	convolution2d_2[0][0]
activation_2 (Activation)	(None, 6, 90, 71)	0	batchnormalization_2[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 6, 22, 17)	0	activation_2[0][0]
flatten_1 (Flatten)	(None, 2244)	0	maxpooling2d_2[0][0]
dense_1 (Dense)	(None, 40)	89760	flatten_1[0][0]
batchnormalization_3 (BatchNormal	(None, 40)	80	dense_1[0][0]
activation_3 (Activation)	(None, 40)	0	batchnormalization_3[0][0]
dense_2 (Dense)	(None, 10)	400	activation_3[0][0]
batchnormalization_4 (BatchNormal	(None, 10)	20	dense_2[0][0]
activation_4 (Activation)	(None, 10)	0	batchnormalization_4[0][0]
dense_3 (Dense)	(None, 2)	22	activation_4[0][0]
activation_5 (Activation)	(None, 2)	0	dense_3[0][0]

Total params: 90522

plot the first image and label from the last batch

```
imshow(X[0,0,:,:],  interpolation='none', origin='lower')
Y[0,:]
```

Plot output of internal layers

following: http://keras.io/getting-started/faq/

do

```
from keras import backend as K
conv0 = K.function([model.layers[0].input], [model.layers[0].output])
out0=conv0([X])[0]
out0.shape
Out[15]: (24, 2, 363, 284)
# this is a batch of 24 samples, there are two channels of outputs, 2 feature maps

# take a look at the two channels
imshow(out0[0,0,:,:])
imshow(out0[0,1,:,:])

# take a look at the final output of the convnet layers,
# since we use batch normalization that behaves differently between training and test,
# we must make this an argument and pass a value when we use the function:
l7fn = K.function([model.layers[0].input, K.learning_phase()],[model.layers[7].output])

convout = l7fn([X,False])[0]
convout.shape
Out[24]: (24, 6, 22, 17)
# look at one of the output channels:
imshow(convout[0,0,:,:])

# take a look at the variables so far
weights = model.weights()
# need to match up with model.summary() to see what is what
# kernel for first convolutional layer
weights[0].shape
(2,1,4,4)
```

Do control+D when done, to quit IPython, model will keep training

Do
git checkout ex02_keras_train.py
to revert changes

Exercises

Explore more of the model

Try different hyper parameter selection in the model

- learning rate
- optimizer momentum
- minibatch size
- validation size
- relu vs. other