Usage of objectives

An objective function (or loss function, or optimization score function) is one of the two parameters required to compile a model:

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

You can either pass the name of an existing objective, or pass a Theano/TensorFlow symbolic function that returns a scalar for each data-point and takes the following two arguments:

- **y_true**: True labels. Theano/TensorFlow tensor.
- **y_pred**: Predictions. Theano/TensorFlow tensor of the same shape as y_true.

The actual optimized objective is the mean of the output array across all datapoints.

For a few examples of such functions, check out the objectives source.

Available objectives

- **mean_squared_error** / **mse**
- **mean_absolute_error** / **mae**
- **mean_absolute_percentage_error** / **mape**
- **mean_squared_logarithmic_error** / **msle**
- **squared_hinge**
- **hinge**
- **binary_crossentropy**: Also known as logloss.
- **categorical_crossentropy**: Also known as multiclass logloss. **Note**: using this objective requires that your labels are binary arrays of shape `(nb_samples, nb_classes)`.
- **sparse_categorical_crossentropy**: As above but accepts sparse labels. **Note**: this objective still requires that your labels have the same number of dimensions as your outputs; you may need to add a length-1 dimension to the shape of your labels, e.g with `np.expand_dims(y, -1)`.
- **kullback_leibler_divergence** / **kld**: Information gain from a predicted probability distribution Q to a true probability distribution P. Gives a measure of difference between both distributions.
- **poisson**: Mean of `(predictions - targets * log(predictions))`
- **cosine_proximity**: The opposite (negative) of the mean cosine proximity between predictions and targets.