Kerasレイヤーを作成

シンプルで状態を持たない独自演算では，`layers.core.Lambda`を用いるべきでしょう． しかし，学習可能な重みを持つ独自演算は，自身でレイヤーを実装する必要があります．

以下に**Keras 2.0**でのレイヤーの枠組みを示します（古いバージョンを使っている場合は，更新してください）． 実装する必要のあるメソッドは3つだけです．

- `build(input_shape)`: これは重みを定義するメソッドです．このメソッドは，`self.built = True`をセットしなければいけません，これは`super([Layer], self).build()`を呼び出しでできます．
- `call(x)`: ここではレイヤーのロジックを記述します．オリジナルのレイヤーでマスキングをサポートしない限り，第1引数である入力テンソルが`call`に渡されることに気を付けてください．
- `compute_output_shape(input_shape)`: 作成したレイヤーの内部で入力のshapeを変更する場合には，ここでshape変換のロジックを指定する必要があります．こうすることでKerasが自動的にshapeを推定します．

```
from keras import backend as K
from keras.engine.topology import Layer
import numpy as np

class MyLayer(Layer):

    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        super(MyLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        # Create a trainable weight variable for this layer.
        self.kernel = self.add_weight(name='kernel',
                                      shape=(input_shape[1], self.output_dim),
                                      initializer='uniform',
                                      trainable=True)
        super(MyLayer, self).build(input_shape)  # Be sure to call this somewhere!

    def call(self, x):
        return K.dot(x, self.kernel)

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.output_dim)
```

既存のKerasレイヤーは何を実装するにしても十分な例を提供しています．なので，躊躇せずソースコードを読んでください！