¡Han encontrado Keras!

Keras: Deep Learning library for Theano and TensorFlow

Decidimos usar esta librería porque:

- Permite realizar un prototipado rápido
- Provee soporte para redes convolucionales, recurrentes y sus combinaciones.
- Funciona tanto en CPU como en GPU (usando Theano o TensorFlow)

Manos a la obra

Para definir y entrenar un modelo neuronal en Keras:

1. Definición.
2. Compilación.
3. Entrenamiento.
4. Evaluación y Predicción.

Esta librería fue diseñada para no requerir más que un par de líneas de código para cada tarea.

Definiendo un modelo secuencial

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_dim=784),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

Definiendo un modelo secuencial

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
...
```

Compilación de un modelo

- Una función de costo o función objetivo.
- Un optimizador.
- Una lista de métricas para monitorear usando los datos de entrenamiento / validación.

```
# para entrenar una clasificación binaria
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])


# para una regresión
model.compile(optimizer='rmsprop', loss='mse')


# descenso por gradiente estocástico
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

¡COMPILAR EL MODELO INCORRECTAMENTE GENERA FALLOS Y RESULTADOS INESPERADOS!

Entrenamiento

```
import numpy as np
data = np.random.random((1000, 784))
labels = np.random.randint(2, size=(1000, 1))
model.fit(data, labels, nb_epoch=150, batch_size=32)
```

```
...
Epoch 143/150
768/768 [==============================] - 0s - loss: 0.4614 - acc: 0.7878
Epoch 144/150
768/768 [==============================] - 0s - loss: 0.4508 - acc: 0.7969
...
```

Evaluación y Predicción

```
import numpy as np
test_data = np.random.random((1000, 784))
test_labels = np.random.randint(2, size=(1000, 1))

print model.evaluate(test_data, test_labels, nb_epoch=10, batch_size=32)
predictions = model.predict(test_data)
```

Documentación oficial:

keras.io