Project: Deep Learning

by Yupeng Wu, Zhifang Zheng, Ze Gong

Created using Julia 1.1.1

In [1]:

```
using Pkg
Pkg.clone("https://github.com/invenia/Keras.jl")

ENV["PYTHON"] = ""

Pkg.build("PyCall")
Pkg.build("Keras")
```

```
  Updating registry at `C:\Users\simon\.julia\registries\General`

┌ Warning: Pkg.clone is only kept for legacy CI script reasons, please use `add`
└ @ Pkg.API C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\usr
\share\julia\stdlib\v1.1\Pkg\src\API.jl:391

  Updating git-repo `https://github.com/JuliaRegistries/General.git`
Fetching: [========================================>]  100.0 %.0 %  Updating gi
t-repo `https://github.com/invenia/Keras.jl`

┌ Info: Assigning UUID 031d7243-eb6a-5793-a290-e3c72503329c to Keras
└ @ Pkg.Types C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\us
r\share\julia\stdlib\v1.1\Pkg\src\Types.jl:841

 Resolving package versions...

┌ Info: Path `C:\Users\simon\.julia\dev\Keras` exists and looks like the correct
package, using existing path
└ @ Pkg.Types C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\us
r\share\julia\stdlib\v1.1\Pkg\src\Types.jl:645

  Updating `C:\Users\simon\.julia\environments\v1.1\Project.toml`
 [no changes]
  Updating `C:\Users\simon\.julia\environments\v1.1\Manifest.toml`
 [no changes]
  Building Keras → `C:\Users\simon\.julia\dev\Keras\deps\build.log`
  Building Conda ──→ `C:\Users\simon\.julia\packages\Conda\kLXeC\deps\build.log
`
  Building PyCall → `C:\Users\simon\.julia\packages\PyCall\ttONZ\deps\build.log
`
  Building Conda ──→ `C:\Users\simon\.julia\packages\Conda\kLXeC\deps\build.log
`
  Building PyCall → `C:\Users\simon\.julia\packages\PyCall\ttONZ\deps\build.log
`
  Building Keras ──→ `C:\Users\simon\.julia\dev\Keras\deps\build.log`
 Resolving package versions...
```

After successfully set the environment, use Pkg.add to add "Keras" and "StatsBase" package to use.

In [2]:

```
Pkg.add("Keras")
using Keras
Pkg.add("StatsBase")
using StatsBase
```

```
 Resolving package versions...
  Updating `C:\Users\simon\.julia\environments\v1.1\Project.toml`
 [no changes]
  Updating `C:\Users\simon\.julia\environments\v1.1\Manifest.toml`
 [no changes]
 Resolving
```

```
Using Theano backend.
WARNING (theano.configdefaults): g++ not available, if using conda: `conda install
m2w64-toolchain`
C:\Users\simon\.julia\conda\3\lib\site-packages\theano\configdefaults.py:560: User
Warning: DeprecationWarning: there is no c++ compiler.This is deprecated and with
Theano 0.11 a c++ compiler will be mandatory
  warnings.warn("DeprecationWarning: there is no c++ compiler."
WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execu
te optimized C-implementations (for both CPU and GPU) and will default to Python i
mplementations. Performance will be severely degraded. To remove this warning, set
Theano flags cxx to an empty string.
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS func
tions.

 package versions...
  Updating `C:\Users\simon\.julia\environments\v1.1\Project.toml`
 [no changes]
  Updating `C:\Users\simon\.julia\environments\v1.1\Manifest.toml`
 [no changes]
```

In [3]:

```
import Keras.Layers: Dense, Activation
```

Now we are able to set the Neural Network model by the Sequential model API.

The Sequential model is a linear stack of layers.

You can create a Sequential model by passing a list of layer instances to the constructor and also simply add layers via the add method.

In [4]:

```
model = Sequential()
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = Sequential() at models.jl:39
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:39
```

Out[4]:

```
Sequential(PyObject <keras.models.Sequential object at 0x00000000012FE940>, Layer
[])
```

The model needs to know what input shape it should expect.

For this reason, the first layer in a Sequential model needs to receive information about its input shape.

(Only the first layer needs to do this, because following layers can do automatic shape inference)

In this case, we are creating a 2D layers by Dense, which support the specification of their input shape via the argument input_dim. Here we set the input_dim with 700 and the input layer with 80 nodes.

In [5]:

```
add!(model, Dense(80, input_dim=700))
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = (::getfield(Keras.Layers, Symbol("##Dense#40#42")))(::Base.Iterator
s.Pairs{Symbol,Int64,Tuple{Symbol},NamedTuple{(:input_dim,),Tuple{Int64}}}, ::Typ
e, ::Int64) at layers.jl:168
└ @ Keras.Layers C:\Users\simon\.julia\dev\Keras\src\layers.jl:168
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = add!(::Sequential, ::Dense) at models.jl:60
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:60
```

We need an activation function to transmit signals down to the next layer.

The activation function of the input layer is Relu. This function is used to "activate" nodes in the hidden layer.

In [6]:

```
add!(model, Activation(:relu))
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = (::getfield(Keras.Layers, Symbol("##Activation#43#45")))(::Base.Iter
ators.Pairs{Union{},Union{},Tuple{},NamedTuple{(),Tuple{}}}, ::Type, ::Symbol) at
layers.jl:168
└ @ Keras.Layers C:\Users\simon\.julia\dev\Keras\src\layers.jl:168
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = add!(::Sequential, ::Activation) at models.jl:60
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:60
```

We set a hidden layer with 10 nodes.

In [7]:

```
add!(model, Dense(10))
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = (::getfield(Keras.Layers, Symbol("##Dense#40#42")))(::Base.Iterator
s.Pairs{Union{},Union{},Tuple{},NamedTuple{(),Tuple{}}}, ::Type, ::Int64) at layer
s.jl:168
└ @ Keras.Layers C:\Users\simon\.julia\dev\Keras\src\layers.jl:168
```

The activation function of the hidden layer is softmax. This function is used to "activate" nodes in the output layer.

In [8]:

```
add!(model, Activation(:softmax))
```

Before training a model, we need to configure the learning process, which is done by the compile method. It receives three arguments:

• An optimizer. This could be the string identifier of an existing optimizer or an instance of the Optimizer class. We use “sgd”, which refers to Stochastic gradient descent optimizer.

• A loss function. This is the objective that the model will try to minimize. Here we use categorical_crossentropy

• A list of metrics. For this classification problem we want to set this to metrics=['accuracy'].

In [9]:

```
compile!(model; loss=:categorical_crossentropy, optimizer=:sgd, metrics=[:accuracy])
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = #compile!#8(::Base.Iterators.Pairs{Symbol,Any,Tuple{Symbol,Symbol,Sy
mbol},NamedTuple{(:loss, :optimizer, :metrics),Tuple{Symbol,Symbol,Array{Symbol,
1}}}}, ::Function, ::Sequential) at models.jl:64
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:64
```

The fit method is what we trains the model for a fixed number of epochs. Epochs are iterations on a dataset. We need to specify a fixed batch size for the inputs when we use the fit method.

Here we set epochs to 100, which means we use 100 epochs to train the model.

And the batch_size is 32, which shows the number of samples per gradient update.

The last parameter is verbose=1, because we want to see the progress bar. If you don’t want see anything, set this to 0.

In [10]:

```
h = fit!(model, rand(10, 700), rand(10, 10); validation_split=0.20, nb_epoch=100, batch_size=32,
verbose=1)
```

```
Train on 8 samples, validate on 2 samples
Epoch 1/100
8/8 [==============================] - 0s - loss: 13.0253 - acc: 0.0000e+00 - val_
loss: 13.3730 - val_acc: 0.0000e+00
Epoch 2/100
8/8 [==============================] - 0s - loss: 12.3513 - acc: 0.1250 - val_los
s: 13.8142 - val_acc: 0.5000
Epoch 3/100
8/8 [==============================] - 0s - loss: 12.0592 - acc: 0.2500 - val_los
s: 13.3732 - val_acc: 0.0000e+00
Epoch 4/100
8/8 [==============================] - 0s - loss: 11.9330 - acc: 0.3750 - val_los
s: 13.5297 - val_acc: 0.5000
Epoch 5/100
8/8 [==============================] - 0s - loss: 11.8625 - acc: 0.3750 - val_los
s: 13.4296 - val_acc: 0.5000
Epoch 6/100
8/8 [==============================] - 0s - loss: 11.8138 - acc: 0.3750 - val_los
s: 13.4939 - val_acc: 0.5000
Epoch 7/100
8/8 [==============================] - 0s - loss: 11.7720 - acc: 0.3750 - val_los
s: 13.4561 - val_acc: 0.5000
Epoch 8/100
8/8 [==============================] - 0s - loss: 11.7369 - acc: 0.3750 - val_los
s: 13.4960 - val_acc: 0.5000
Epoch 9/100
8/8 [==============================] - 0s - loss: 11.7043 - acc: 0.3750 - val_los
s: 13.4806 - val_acc: 0.5000
Epoch 10/100
8/8 [==============================] - 0s - loss: 11.6756 - acc: 0.3750 - val_los
s: 13.5014 - val_acc: 0.5000
Epoch 11/100
8/8 [==============================] - 0s - loss: 11.6489 - acc: 0.3750 - val_los
s: 13.4913 - val_acc: 0.5000
Epoch 12/100
8/8 [==============================] - 0s - loss: 11.6251 - acc: 0.5000 - val_los
s: 13.4976 - val_acc: 0.5000
Epoch 13/100
8/8 [==============================] - 0s - loss: 11.6034 - acc: 0.5000 - val_los
s: 13.5082 - val_acc: 0.5000
Epoch 14/100
8/8 [==============================] - 0s - loss: 11.5837 - acc: 0.5000 - val_los
s: 13.5067 - val_acc: 0.5000
Epoch 15/100
8/8 [==============================] - 0s - loss: 11.5649 - acc: 0.5000 - val_los
s: 13.5195 - val_acc: 0.5000
Epoch 16/100
8/8 [==============================] - 0s - loss: 11.5475 - acc: 0.5000 - val_los
s: 13.5165 - val_acc: 0.0000e+00
Epoch 17/100
8/8 [==============================] - 0s - loss: 11.5311 - acc: 0.5000 - val_los
s: 13.5178 - val_acc: 0.0000e+00
Epoch 18/100
8/8 [==============================] - 0s - loss: 11.5161 - acc: 0.5000 - val_los
s: 13.5205 - val_acc: 0.0000e+00
Epoch 19/100
8/8 [==============================] - 0s - loss: 11.5017 - acc: 0.5000 - val_los
s: 13.5351 - val_acc: 0.0000e+00
Epoch 20/100
8/8 [==============================] - 0s - loss: 11.4883 - acc: 0.5000 - val_los
s: 13.5183 - val_acc: 0.0000e+00
```

```
Epoch 21/100
8/8 [==============================] - 0s - loss: 11.4754 - acc: 0.5000 - val_los
s: 13.5398 - val_acc: 0.5000
Epoch 22/100
8/8 [==============================] - 0s - loss: 11.4628 - acc: 0.5000 - val_los
s: 13.5308 - val_acc: 0.0000e+00
Epoch 23/100
8/8 [==============================] - 0s - loss: 11.4512 - acc: 0.5000 - val_los
s: 13.5516 - val_acc: 0.0000e+00
Epoch 24/100
8/8 [==============================] - 0s - loss: 11.4411 - acc: 0.5000 - val_los
s: 13.5409 - val_acc: 0.0000e+00
Epoch 25/100
8/8 [==============================] - 0s - loss: 11.4318 - acc: 0.5000 - val_los
s: 13.5530 - val_acc: 0.0000e+00
Epoch 26/100
8/8 [==============================] - 0s - loss: 11.4223 - acc: 0.5000 - val_los
s: 13.5500 - val_acc: 0.0000e+00
Epoch 27/100
8/8 [==============================] - 0s - loss: 11.4139 - acc: 0.5000 - val_los
s: 13.5634 - val_acc: 0.0000e+00
Epoch 28/100
8/8 [==============================] - 0s - loss: 11.4060 - acc: 0.5000 - val_los
s: 13.5546 - val_acc: 0.0000e+00
Epoch 29/100
8/8 [==============================] - 0s - loss: 11.3979 - acc: 0.5000 - val_los
s: 13.5727 - val_acc: 0.0000e+00
Epoch 30/100
8/8 [==============================] - 0s - loss: 11.3906 - acc: 0.5000 - val_los
s: 13.5622 - val_acc: 0.0000e+00
Epoch 31/100
8/8 [==============================] - 0s - loss: 11.3832 - acc: 0.5000 - val_los
s: 13.5775 - val_acc: 0.0000e+00
Epoch 32/100
8/8 [==============================] - 0s - loss: 11.3757 - acc: 0.6250 - val_los
s: 13.5721 - val_acc: 0.0000e+00
Epoch 33/100
8/8 [==============================] - 0s - loss: 11.3688 - acc: 0.6250 - val_los
s: 13.5886 - val_acc: 0.0000e+00
Epoch 34/100
8/8 [==============================] - 0s - loss: 11.3621 - acc: 0.6250 - val_los
s: 13.5815 - val_acc: 0.0000e+00
Epoch 35/100
8/8 [==============================] - 0s - loss: 11.3558 - acc: 0.6250 - val_los
s: 13.6035 - val_acc: 0.0000e+00
Epoch 36/100
8/8 [==============================] - 0s - loss: 11.3501 - acc: 0.6250 - val_los
s: 13.5932 - val_acc: 0.0000e+00
Epoch 37/100
8/8 [==============================] - 0s - loss: 11.3443 - acc: 0.6250 - val_los
s: 13.6026 - val_acc: 0.0000e+00
Epoch 38/100
8/8 [==============================] - 0s - loss: 11.3395 - acc: 0.6250 - val_los
s: 13.6002 - val_acc: 0.0000e+00
Epoch 39/100
8/8 [==============================] - 0s - loss: 11.3346 - acc: 0.6250 - val_los
s: 13.6208 - val_acc: 0.0000e+00
Epoch 40/100
8/8 [==============================] - 0s - loss: 11.3299 - acc: 0.6250 - val_los
s: 13.6007 - val_acc: 0.0000e+00
Epoch 41/100
```

```
8/8 [==============================] - 0s - loss: 11.3255 - acc: 0.6250 - val_los
s: 13.6267 - val_acc: 0.0000e+00
Epoch 42/100
8/8 [==============================] - 0s - loss: 11.3213 - acc: 0.6250 - val_los
s: 13.6189 - val_acc: 0.0000e+00
Epoch 43/100
8/8 [==============================] - 0s - loss: 11.3168 - acc: 0.6250 - val_los
s: 13.6338 - val_acc: 0.0000e+00
Epoch 44/100
8/8 [==============================] - 0s - loss: 11.3132 - acc: 0.6250 - val_los
s: 13.6216 - val_acc: 0.0000e+00
Epoch 45/100
8/8 [==============================] - 0s - loss: 11.3094 - acc: 0.6250 - val_los
s: 13.6389 - val_acc: 0.0000e+00
Epoch 46/100
8/8 [==============================] - 0s - loss: 11.3057 - acc: 0.6250 - val_los
s: 13.6284 - val_acc: 0.0000e+00
Epoch 47/100
8/8 [==============================] - 0s - loss: 11.3024 - acc: 0.6250 - val_los
s: 13.6483 - val_acc: 0.0000e+00
Epoch 48/100
8/8 [==============================] - 0s - loss: 11.2988 - acc: 0.6250 - val_los
s: 13.6351 - val_acc: 0.0000e+00
Epoch 49/100
8/8 [==============================] - 0s - loss: 11.2954 - acc: 0.7500 - val_los
s: 13.6499 - val_acc: 0.0000e+00
Epoch 50/100
8/8 [==============================] - 0s - loss: 11.2923 - acc: 0.6250 - val_los
s: 13.6394 - val_acc: 0.0000e+00
Epoch 51/100
8/8 [==============================] - 0s - loss: 11.2896 - acc: 0.6250 - val_los
s: 13.6597 - val_acc: 0.0000e+00
Epoch 52/100
8/8 [==============================] - 0s - loss: 11.2864 - acc: 0.8750 - val_los
s: 13.6443 - val_acc: 0.0000e+00
Epoch 53/100
8/8 [==============================] - 0s - loss: 11.2837 - acc: 0.7500 - val_los
s: 13.6602 - val_acc: 0.0000e+00
Epoch 54/100
8/8 [==============================] - 0s - loss: 11.2809 - acc: 0.6250 - val_los
s: 13.6528 - val_acc: 0.0000e+00
Epoch 55/100
8/8 [==============================] - 0s - loss: 11.2782 - acc: 0.7500 - val_los
s: 13.6665 - val_acc: 0.0000e+00
Epoch 56/100
8/8 [==============================] - 0s - loss: 11.2756 - acc: 0.8750 - val_los
s: 13.6603 - val_acc: 0.0000e+00
Epoch 57/100
8/8 [==============================] - 0s - loss: 11.2733 - acc: 0.7500 - val_los
s: 13.6658 - val_acc: 0.0000e+00
Epoch 58/100
8/8 [==============================] - 0s - loss: 11.2709 - acc: 0.7500 - val_los
s: 13.6687 - val_acc: 0.0000e+00
Epoch 59/100
8/8 [==============================] - 0s - loss: 11.2686 - acc: 0.8750 - val_los
s: 13.6692 - val_acc: 0.0000e+00
Epoch 60/100
8/8 [==============================] - 0s - loss: 11.2664 - acc: 0.8750 - val_los
s: 13.6744 - val_acc: 0.0000e+00
Epoch 61/100
8/8 [==============================] - 0s - loss: 11.2643 - acc: 0.8750 - val_los
```

s: 13.6743 - val_acc: 0.0000e+00
Epoch 62/100
8/8 [==============================] - 0s - loss: 11.2622 - acc: 0.8750 - val_los
s: 13.6759 - val_acc: 0.0000e+00
Epoch 63/100
8/8 [==============================] - 0s - loss: 11.2601 - acc: 0.8750 - val_los
s: 13.6779 - val_acc: 0.0000e+00
Epoch 64/100
8/8 [==============================] - 0s - loss: 11.2581 - acc: 0.8750 - val_los
s: 13.6796 - val_acc: 0.0000e+00
Epoch 65/100
8/8 [==============================] - 0s - loss: 11.2563 - acc: 0.8750 - val_los
s: 13.6849 - val_acc: 0.0000e+00
Epoch 66/100
8/8 [==============================] - 0s - loss: 11.2544 - acc: 0.8750 - val_los
s: 13.6801 - val_acc: 0.0000e+00
Epoch 67/100
8/8 [==============================] - 0s - loss: 11.2526 - acc: 0.8750 - val_los
s: 13.6869 - val_acc: 0.0000e+00
Epoch 68/100
8/8 [==============================] - 0s - loss: 11.2510 - acc: 0.8750 - val_los
s: 13.6865 - val_acc: 0.0000e+00
Epoch 69/100
8/8 [==============================] - 0s - loss: 11.2492 - acc: 0.8750 - val_los
s: 13.6917 - val_acc: 0.0000e+00
Epoch 70/100
8/8 [==============================] - 0s - loss: 11.2477 - acc: 0.8750 - val_los
s: 13.6881 - val_acc: 0.0000e+00
Epoch 71/100
8/8 [==============================] - 0s - loss: 11.2462 - acc: 0.8750 - val_los
s: 13.6968 - val_acc: 0.0000e+00
Epoch 72/100
8/8 [==============================] - 0s - loss: 11.2447 - acc: 0.8750 - val_los
s: 13.6939 - val_acc: 0.0000e+00
Epoch 73/100
8/8 [==============================] - 0s - loss: 11.2431 - acc: 0.8750 - val_los
s: 13.7007 - val_acc: 0.0000e+00
Epoch 74/100
8/8 [==============================] - 0s - loss: 11.2417 - acc: 0.8750 - val_los
s: 13.6934 - val_acc: 0.0000e+00
Epoch 75/100
8/8 [==============================] - 0s - loss: 11.2404 - acc: 0.8750 - val_los
s: 13.7039 - val_acc: 0.0000e+00
Epoch 76/100
8/8 [==============================] - 0s - loss: 11.2389 - acc: 0.8750 - val_los
s: 13.7018 - val_acc: 0.0000e+00
Epoch 77/100
8/8 [==============================] - 0s - loss: 11.2377 - acc: 0.8750 - val_los
s: 13.7034 - val_acc: 0.0000e+00
Epoch 78/100
8/8 [==============================] - 0s - loss: 11.2365 - acc: 0.8750 - val_los
s: 13.6990 - val_acc: 0.0000e+00
Epoch 79/100
8/8 [==============================] - 0s - loss: 11.2352 - acc: 0.8750 - val_los
s: 13.7102 - val_acc: 0.0000e+00
Epoch 80/100
8/8 [==============================] - 0s - loss: 11.2341 - acc: 0.8750 - val_los
s: 13.7036 - val_acc: 0.0000e+00
Epoch 81/100
8/8 [==============================] - 0s - loss: 11.2329 - acc: 0.8750 - val_los
s: 13.7106 - val_acc: 0.0000e+00

```
Epoch 82/100
8/8 [==============================] - 0s - loss: 11.2317 - acc: 0.8750 - val_los
s: 13.7084 - val_acc: 0.0000e+00
Epoch 83/100
8/8 [==============================] - 0s - loss: 11.2306 - acc: 0.8750 - val_los
s: 13.7147 - val_acc: 0.0000e+00
Epoch 84/100
8/8 [==============================] - 0s - loss: 11.2296 - acc: 0.8750 - val_los
s: 13.7113 - val_acc: 0.0000e+00
Epoch 85/100
8/8 [==============================] - 0s - loss: 11.2285 - acc: 0.8750 - val_los
s: 13.7165 - val_acc: 0.0000e+00
Epoch 86/100
8/8 [==============================] - 0s - loss: 11.2275 - acc: 0.8750 - val_los
s: 13.7172 - val_acc: 0.0000e+00
Epoch 87/100
8/8 [==============================] - 0s - loss: 11.2265 - acc: 0.8750 - val_los
s: 13.7186 - val_acc: 0.0000e+00
Epoch 88/100
8/8 [==============================] - 0s - loss: 11.2255 - acc: 0.8750 - val_los
s: 13.7198 - val_acc: 0.0000e+00
Epoch 89/100
8/8 [==============================] - 0s - loss: 11.2246 - acc: 0.8750 - val_los
s: 13.7233 - val_acc: 0.0000e+00
Epoch 90/100
8/8 [==============================] - 0s - loss: 11.2237 - acc: 0.8750 - val_los
s: 13.7209 - val_acc: 0.0000e+00
Epoch 91/100
8/8 [==============================] - 0s - loss: 11.2228 - acc: 0.8750 - val_los
s: 13.7283 - val_acc: 0.0000e+00
Epoch 92/100
8/8 [==============================] - 0s - loss: 11.2219 - acc: 0.8750 - val_los
s: 13.7296 - val_acc: 0.0000e+00
Epoch 93/100
8/8 [==============================] - 0s - loss: 11.2210 - acc: 0.8750 - val_los
s: 13.7296 - val_acc: 0.0000e+00
Epoch 94/100
8/8 [==============================] - 0s - loss: 11.2202 - acc: 0.8750 - val_los
s: 13.7321 - val_acc: 0.0000e+00
Epoch 95/100
8/8 [==============================] - 0s - loss: 11.2195 - acc: 0.8750 - val_los
s: 13.7333 - val_acc: 0.0000e+00
Epoch 96/100
8/8 [==============================] - 0s - loss: 11.2186 - acc: 0.8750 - val_los
s: 13.7372 - val_acc: 0.0000e+00
Epoch 97/100
8/8 [==============================] - 0s - loss: 11.2178 - acc: 0.8750 - val_los
s: 13.7377 - val_acc: 0.0000e+00
Epoch 98/100
8/8 [==============================] - 0s - loss: 11.2171 - acc: 0.8750 - val_los
s: 13.7397 - val_acc: 0.0000e+00
Epoch 99/100
8/8 [==============================] - 0s - loss: 11.2163 - acc: 0.8750 - val_los
s: 13.7371 - val_acc: 0.0000e+00
Epoch 100/100
8/8 [==============================] - 0s - loss: 11.2156 - acc: 0.8750 - val_los
s: 13.7437 - val_acc: 0.0000e+00
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = #fit!#9(::Int64, ::Base.Iterators.Pairs{Symbol,Real,Tuple{Symbol,Sym
bol,Symbol},NamedTuple{(:validation_split, :nb_epoch, :batch_size),Tuple{Float64,I
nt64,Int64}}}, ::Function, ::Sequential, ::Array{Float64,2}, ::Vararg{Array{Float6
4,2},N} where N) at models.jl:68
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:68
C:\Users\simon\.julia\conda\3\lib\site-packages\keras\models.py:826: UserWarning:
The `nb_epoch` argument in `fit` has been renamed `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
```

Out[10]:

```
PyObject <keras.callbacks.History object at 0x00000000393C6B00>
```

Overfitting always occurs when our model becomes really good at classifying or predicting on data that was included in the training set. But it is not good at classifying data that isn't trained on.

We ca know overfitting based on metrics that are given for our training and validation data during the training process.

If the validation accuracy worse than training accuracy, then our model is overfitting.

We also get overfitting if the models metrics were good but when we use the model to predict on test data, it is not accurately classifying the data in the test set.

In the test mode, we need input data and target data, and the calculation is carried out in batches.

Then it returns to the error value (loss) and the evaluation standard value.

We can get the evaluation standard value is about 0.2, which is small and close to 0.

In [11]:

```
evaluate(model, rand(10, 700), rand(10, 10); batch_size=5, verbose=1)
```

```
┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = #evaluate#10(::Int64, ::Int64, ::Function, ::Sequential, ::Array{Flo
at64,2}, ::Array{Float64,2}) at models.jl:72
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:72

 5/10 [==============>..............] - ETA: 0s
```

Out[11]:

```
2-element Array{Float64,1}:
 11.976428508758545
  0.0
```

Then we can use this model to predict on other dataset.

In [12]:

```
predict(model, rand(10, 700); batch_size=5, verbose=1)
```

```
 5/10 [==============>..............] - ETA: 0s

┌ Warning: `getindex(o::PyObject, s::Symbol)` is deprecated in favor of dot overl
oading (`getproperty`) so elements should now be accessed as e.g. `o.s` instead of
`o[:s]`.
│   caller = #predict#11(::Int64, ::Int64, ::Function, ::Sequential, ::Array{Floa
t64,2}) at models.jl:76
└ @ Keras C:\Users\simon\.julia\dev\Keras\src\models.jl:76
```

Out[12]:

```
10×10 Array{Float32,2}:
 0.0987271  0.133345   0.118539   …  0.0968778  0.0635645  0.0530889
 0.0641723  0.0983444  0.0940889     0.172037   0.107849   0.0709795
 0.0876788  0.0875767  0.0986511     0.168873   0.0646441  0.0912005
 0.138393   0.124779   0.0870739     0.110914   0.135138   0.136531
 0.0708688  0.0960495  0.0594764     0.119417   0.0838852  0.142463
 0.0663076  0.105571   0.0610621  …  0.0827518  0.157182   0.072068
 0.106416   0.100854   0.0843876     0.0752178  0.105821   0.0441032
 0.0711867  0.162014   0.0966662     0.0492741  0.207805   0.0866259
 0.0878673  0.165861   0.142253      0.0647497  0.159111   0.0748197
 0.0566073  0.064067   0.09489       0.124726   0.0966054  0.0993727
```

Reference:

https://keras.io/models/ (https://keras.io/models/)

https://github.com/invenia/Keras.jl (https://github.com/invenia/Keras.jl)

Youtube Link:

https://youtu.be/9_lLdjXJu8E (https://youtu.be/9_lLdjXJu8E)

In [ ]:

```

```