Simple neural network classification over MNIST

Anton Antonov
MathematicaVsR at GitHub
May 2018

Introduction

This notebook is part of the MathematicaVsR at GitHub project "DeepLearningExamples".

This notebook has code corresponding to neural network creation, training, and testing in RStudio's opening Keras page: https://keras.rstudio.com/index.html .

(Also we can say that this notebook has code that corresponds to code in the book "Deep learning with R" by F. Chollet and J. J. Allaire. See the GitHub repository: https://github.com/jjallaire/deep-learning-with-r-notebooks ; specifically the notebook "A first look at a neural network".)

Get code

Here we load a package with utilities:

```
In[1]:= Import["https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/MathematicaForPredictionUtilities.m"]
```

Get data

We can get the data from Wolfram's data repository:

```
In[2]:= (*ResourceObject["MNIST"]*)
```

or we can directly use `ExampleData`:

```
In[3]:= trainingData = ExampleData[{"MachineLearning", "MNIST"}, "TrainingData"];
testData = ExampleData[{"MachineLearning", "MNIST"}, "TestData"];
```

Here is a description of the variable names:

```
In[5]:= colNames = Flatten[List @@ ExampleData[{"MachineLearning", "MNIST"}, "VariableDescriptions"]];
GridTableForm[List /@ colNames]
```

Out[6]=

#	1
1	28x28 grayscale image
2	Integer in the range 1-10

Data summaries

```
In[7]:= Dimensions[Flatten@*List @@@ trainingData]
Out[7]= {60 000, 2}
```

```
In[8]:= Dimensions[Flatten@*List @@@ testData]
Out[8]= {10 000, 2}
```

In[9]:= `Magnify[RecordsSummary[trainingData, Thread → True], 0.8]`

Out[9]=

1 column 1	
	1
	1
	1
	1
	1
	1
(Other)	59 994

→

1 column 1	
Min	0
1st Qu	2
Median	4
Mean	4.45393
3rd Qu	7
Max	9

In[10]:= `Magnify[RecordsSummary[testData, Thread → True], 0.8]`

Out[10]=

1 column 1	
	1
	1
	1
	1
	1
	1
(Other)	9994

→

1 column 1	
Min	0
1st Qu	2
Median	4
Mean	4.4434
3rd Qu	7
Max	9

In[11]:= `Tally[ImageDimensions /@ trainingData[[All, 1]]]`

Out[11]= `{{{28, 28}, 60 000}}`

Transform data: images into 1D arrays

In[12]:= `trainingData = Map[Flatten[ImageData[#[[1]]]] → #[[2]] &, trainingData];`

In[13]:= `testData = Map[Flatten[ImageData[#[[1]]]] → #[[2]] &, testData];`

The neural network

Build the network:

In[14]:=
```
model =
 NetChain[{
   LinearLayer[256, "Input" → Length[First@trainingData[[All, 1]]]],
   ElementwiseLayer[Ramp],
   DropoutLayer[0.4],
   LinearLayer[128],
   ElementwiseLayer[Ramp],
   DropoutLayer[0.3],
   LinearLayer[10],
   SoftmaxLayer["Output" -> NetDecoder[{"Class", Range[0, 9]}]]
  }]
```

Out[14]= NetChain[uninitialized | Input port: vector (size: 784) | Output port: class | Number of layers: 8]

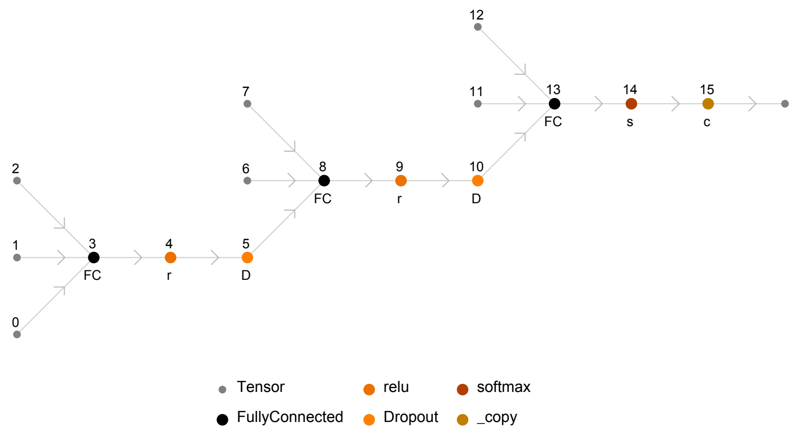
From the documentation:

CrossEntropyLossLayer["Index"] is used automatically by NetTrain when the final activation used for an output is a SoftmaxLayer.

Visualize the network:

In[15]:=
```
NetInformation[model, "MXNetNodeGraphPlot"]
```

Out[15]=



Train the network:

In[16]:=
```
tNet = NetTrain[model, trainingData, ValidationSet → Scaled[0.05], "MaxTrainingRounds" → 80]
```

Out[16]= NetChain[Input port: vector (size: 784) | Output port: class | Number of layers: 8]

Testing

Predict results for the test data:

In[17]:=
```
clRes = tNet /@ testData⟦All, 1⟧;
Short[clRes]
```

Out[18]//Short=
```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, <<9962>>, 4, 9, 9, 9, 9, 4, 9, 9, 9, 5, 4, 4, 9, 9, 9, 9, 9, 9, 4}
```

This computes a classifier measurements object:

In[19]:=
```
cm = ClassifierMeasurements[tNet, trainingData]
```

Out[19]=
```
ClassifierMeasurementsObject[ Classifier: Net
                              Number of test examples: 60000
                              Data not in notebook; Store now » ]
```

Here is the accuracy:

In[20]:=
```
cm["Accuracy"]
```

Out[20]=
```
0.937617
```

Here are the precision, recall, specificity, false positive rate metrics:

In[21]:=
```
ms = {"Precision", "Recall", "Specificity", "FalsePositiveRate"};
Transpose[Dataset[AssociationThread[ms -> cm[ms]]]]
```

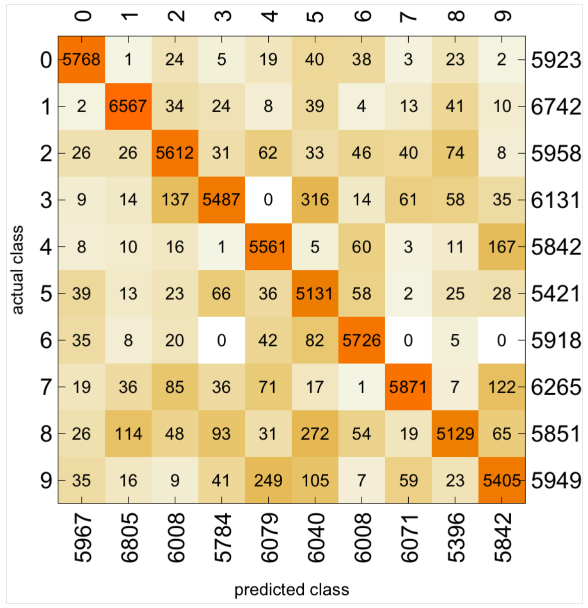
Out[22]=

	Precision	Recall	Specificity	FalsePositiveRate
0	0.96665	0.973831	0.99632	0.00367994
1	0.965026	0.974043	0.995531	0.00446881
2	0.934088	0.941927	0.992672	0.00732763
3	0.948651	0.89496	0.994487	0.00551338
4	0.914789	0.9519	0.990435	0.00956461
5	0.849503	0.946504	0.983345	0.0166548
6	0.953063	0.967557	0.994786	0.0052143
7	0.967056	0.937111	0.996278	0.00372197
8	0.950519	0.876602	0.995069	0.00493084
9	0.925197	0.908556	0.991915	0.00808496

Here is a confusion matrix plot:

In[23]:=
```
cm["ConfusionMatrixPlot"]
```

Out[23]=



Here are the top confused digits:

In[24]:=
```
cm["TopConfusions"]
```

Out[24]= {3 → 5, 4 → 9, 7 → 9, 2 → 8, 8 → 9, 2 → 4, 3 → 7, 4 → 6, 3 → 8, 5 → 6}