Predicting house prices: a regression example

Anton Antonov
MathematicaVsR at GitHub
May 2018

Introduction

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

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 “Predicting house prices: a regression example”.

Get code

Here we load a package with utilities:

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

» Importing from GitHub: MosaicPlot.m

» Importing from GitHub: CrossTabulate.m

Get data

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

In[2]:=
```
(*ResourceObject["Sample Data: Boston Homes"]*)
```

or we can directly use ExampleData:

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

Here is a description of the variable names:

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

Out[6]=

#	1
1	Per capita crime rate by town
2	Proportion of residential land zoned for lots over 25000 square feet
3	Proportion of non-retail business acres per town
4	Charles River dummy variable (1 if tract bounds river, 0 otherwise)
5	Nitrogen oxide concentration (parts per 10 million)
6	Average number of rooms per dwelling
7	Proportion of owner-occupied units built prior to 1940
8	Weighted mean of distances to five Boston employment centers
9	Index of accessibility to radial highways
10	Full-value property-tax rater per $10000
11	Pupil-teacher ratio by town
12	1000(Bk-0.63)^2 where Bk is the proportion of blacks by town
13	Lower status of the population (percent)
14	Median value of owner-occupied homes in $1000s

Data summaries

In[7]:=
```
Dimensions[Flatten@*List @@@ trainingData]
```

Out[7]= {338, 14}

In[8]:=
```
Dimensions[Flatten@*List @@@ testData]
```

Out[8]= {168, 14}

In[9]:=
```
Magnify[RecordsSummary[trainingData, Thread → True], 0.6]
```

Out[9]=

1 column 1		2 column 2		3 column 3		4 column 4		5 column 5		6 column 6		7 column 7		8 column 8		9 column 9		10 column 10		11 column 11		12 column 12		13 column 13		1 column 1	
Min	0.00906	1st Qu	0	Min	0.46	1st Qu	0	Min	0.385	Min	3.561	Min	2.9	Min	1.137	Min	1	Min	187	Min	12.6	Min	2.52	Min	1.73	Min	5
1st Qu	0.08873	Median	0	1st Qu	5.19	3rd Qu	0	1st Qu	0.453	1st Qu	5.888	1st Qu	45	1st Qu	2.1099	1st Qu	4	1st Qu	277	1st Qu	17.4	Mean	358.954	1st Qu	6.93	1st Qu	17.2
Median	0.274475	Min	0	Median	8.56	Median	0	Median	0.538	Median	6.209	Mean	69.1896	Median	3.2157	Median	5	Median	315	Mean	18.4808	1st Qu	376.14	Median	11.645	Median	21.3
3rd Qu	2.37857	Mean	10.9734	Mean	10.8893	Min	0	Mean	0.551747	Mean	6.31367	Median	79.05	Mean	3.786	3rd Qu	8	Mean	400.757	Median	19.1	Median	391.52	Mean	12.5943	Mean	22.7145
Mean	3.50511	3rd Qu	12.5	3rd Qu	18.1	Mean	0.0680473	3rd Qu	0.614	3rd Qu	6.683	3rd Qu	94	3rd Qu	4.9671	Mean	9.24556	3rd Qu	666	3rd Qu	20.2	3rd Qu	396.33	3rd Qu	16.96	3rd Qu	26.4
Max	88.9762	Max	100	Max	27.74	Max	1	Max	0.871	Max	8.725	Max	100	Max	12.1265	Max	24	Max	711	Max	22	Max	396.9	Max	37.97	Max	50

In[10]:=
```
Magnify[RecordsSummary[testData, Thread → True], 0.6]
```

Out[10]=

1 column 1		2 column 2		3 column 3		4 column 4		5 column 5		6 column 6		7 column 7		8 column 8		9 column 9		10 column 10		11 column 11		12 column 12		13 column 13		1 column 1	
Min	0.00632	1st Qu	0	Min	1.21	1st Qu	0	Min	0.394	Min	3.863	Min	6.2	Min	1.1296	Min	1	Min	188	Min	12.6	Min	0.32	Min	2.88	Min	8.4
1st Qu	0.068935	Median	0	1st Qu	4.93	3rd Qu	0	1st Qu	0.446	1st Qu	5.8765	1st Qu	45.35	1st Qu	1.96175	1st Qu	4	1st Qu	285.5	1st Qu	16.6	Mean	352.088	1st Qu	7.06	1st Qu	16.1
Median	0.21693	Min	0	Median	10.59	Median	0	Median	0.538	Median	6.2005	Mean	67.3381	Median	3.1423	Median	5	Median	377	Mean	18.4048	1st Qu	370.5	Median	11.25	Median	20.95
Mean	3.83163	Mean	12.1488	Mean	11.6348	Min	0	Mean	0.560626	Mean	6.22622	Median	72.95	Mean	3.81324	Mean	10.1607	Mean	423.286	Median	18.95	Median	391.44	Mean	12.7714	Mean	22.1673
3rd Qu	4.75225	3rd Qu	12.5	3rd Qu	18.1	Mean	0.0714286	3rd Qu	0.668	3rd Qu	6.543	3rd Qu	94.25	3rd Qu	5.4007	3rd Qu	24	3rd Qu	666	3rd Qu	20.2	3rd Qu	395.69	3rd Qu	16.92	3rd Qu	24.75
Max	73.5341	Max	95	Max	27.74	Max	1	Max	0.871	Max	8.78	Max	100	Max	10.7103	Max	24	Max	711	Max	21.2	Max	396.9	Max	34.77	Max	50

Standardize data

First we turn the data sets into arrays:

In[12]:=
```
trainingData = Flatten@*List @@@ trainingData;
testData = Flatten@*List @@@ testData;
```

Note that the standardisation of the data is done by using means and standard deviations derived from the training data only.

In[14]:=
```
means = Mean[trainingData];
sds = N[StandardDeviation /@ Transpose[trainingData]];
```

In[16]:=
```
trainingData = Transpose@MapThread[(#1 - #2) / #3 &, {Transpose[trainingData], means, sds}];
testData = Transpose@MapThread[(#1 - #2) / #3 &, {Transpose[testData], means, sds}];
```

Here we transform the data back into lists of `example→label` rules:

In[18]:=
```
trainingData = Map[Most[#] → Last[#] &, trainingData];
testData = Map[Most[#] → Last[#] &, testData];
```

In[20]:=
```
Magnify[RecordsSummary[trainingData, Thread → True], 0.6]
```

Out[20]=

1 column 1	
Min	−0.397501
1st Qu	−0.388442
Median	−0.367323
3rd Qu	−0.128088
Mean	-3.67885×10^{-17}
Max	9.71804

2 column 2	
1st Qu	−0.490044
Median	−0.490044
Min	−0.490044
Mean	-4.66425×10^{-17}
3rd Qu	0.0681755
Max	3.97571

3 column 3	
Min	−1.54121
1st Qu	−0.842221
Median	−0.344212
Mean	1.64234×10^{-16}
3rd Qu	1.06558
Max	2.49016

4 column 4	
1st Qu	−0.269814
3rd Qu	−0.269814
Median	−0.269814
Min	−0.269814
Mean	-1.0511×10^{-17}
Max	3.69529

5 column 5	
Min	−1.50558
1st Qu	−0.891603
Median	−0.124127
Mean	-3.33724×10^{-16}
3rd Qu	0.562088
Max	2.88258

6 column 6	
Min	−3.90689
1st Qu	−0.604155
Median	−0.148557
Mean	-2.49636×10^{-16}
3rd Qu	0.524195
Max	3.42242

7 column 7	
Min	−2.36929
1st Qu	−0.864572
Mean	2.9168×10^{-16}
Median	0.352423
3rd Qu	0.886758
Max	1.10121

8 column 8	
Min	−1.26549
1st Qu	−0.800712
Median	−0.272445
Mean	-2.21388×10^{-16}
3rd Qu	0.564241
Max	3.98446

9 column 9	
Min	−0.970697
1st Qu	−0.617526
Median	−0.499803
3rd Qu	−0.146632
Mean	-1.97081×10^{-17}
Max	1.73694

10 column 10	
Min	−1.28269
1st Qu	−0.742631
Median	−0.514605
Mean	-1.83942×10^{-17}
3rd Qu	1.59164
Max	1.86167

11 column 11	
Min	−2.67927
1st Qu	−0.492397
Mean	2.28614×10^{-16}
Median	0.282121
3rd Qu	0.78328
Max	1.60336

12 column 12	
Min	−4.00028
Mean	4.28323×10^{-16}
1st Qu	0.192884
Median	0.365495
3rd Qu	0.419478
Max	0.425875

13 column 13	
Min	−1.53127
1st Qu	−0.79835
Median	−0.133794
Mean	1.64234×10^{-16}
3rd Qu	0.61533
Max	3.57659

→

1 column 1	
Min	−1.90526
1st Qu	−0.593105
Median	−0.152134
Mean	1.70804×10^{-16}
3rd Qu	0.39639
Max	2.93466

The neural network

Build the network:

In[21]:=
```
model =
 NetChain[{
   LinearLayer[64, "Input" → Dimensions[trainingData[[All, 1]][[2]]],
   ElementwiseLayer[Ramp],
   LinearLayer[64],
   ElementwiseLayer[Ramp],
   LinearLayer[1]
  }, "Output" → "Scalar"]
```

Out[21]=
NetChain[uninitialized

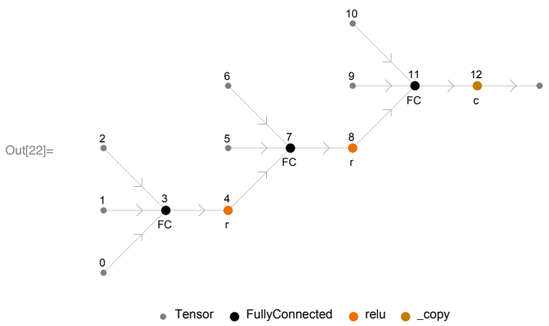
	Input	vector (size: 13)
1	LinearLayer	vector (size: 64)
2	Ramp	vector (size: 64)
3	LinearLayer	vector (size: 64)
4	Ramp	vector (size: 64)
5	LinearLayer	vector (size: 1)
	Output	scalar

]

Visualize the network:

In[22]:=
```
NetInformation[model, "MXNetNodeGraphPlot"]
```

Out[22]=



Train the network:

In[23]:=
```
tNet = NetTrain[model, trainingData,
  ValidationSet → Scaled[.05], "MaxTrainingRounds" → 80]
```

Out[23]=
NetChain[

	Input	vector (size: 13)
1	LinearLayer	vector (size: 64)
2	Ramp	vector (size: 64)
3	LinearLayer	vector (size: 64)
4	Ramp	vector (size: 64)
5	LinearLayer	vector (size: 1)
	Output	scalar

]

Testing

Predict results for the test data:

In[32]:=
```
pRes = tNet /@ testData[[All, 1]];
Short[pRes]
```

Out[33]//Short=
```
{0.588873, 0.909774, <<164>>, -0.952313, -0.37624}
```

Mean absolute error:

In[34]:=
```
Mean[Abs[pRes - (Last /@ testData)]]
```

Out[34]=
```
0.26433
```

Root mean square error:

In[35]:=
```
RootMeanSquare[pRes - (Last /@ testData)]
```

Out[35]=
```
0.451546
```

Compare the predicted results with the actual values using a scatter plot:

In[36]:=
```
Show[
 ListPlot[Transpose[{pRes, Last /@ testData}], FrameLabel → {"Predicted", "Actual"},
  PlotTheme → "Detailed", PlotRange → All, ImageSize → Large],
 Plot[{x, x}, {x, Min[pRes], Max[pRes]}]]
```

Out[36]=