Training neural networks - GRB byte histogram classification

Full training set:

11243 examples

Positive examples:

4000 positive examples collected from <ftp://hydro1.sci.gsfc.nasa.gov/data>, and

243 positive examples from the following

[71] <http://www.globalmarinenet.com/grib_downloads.php#Download>

[80] <ftp://nomads.ncdc.noaa.gov/CFSR/HP_monthly_means/archive/>

[79] <ftp://nomads.ncdc.noaa.gov/CFSR/HP_monthly_means/archive/>

[13] <ftp://nomads.ncdc.noaa.gov/RUC/20km/archive/> )

Negative examples:

7000 negative examples collected from <http://digitalcorpora.org/corpora/govdocs>

Used training set:

[3000 -11243] = 8243 examples (1000 from the 4000 positive examples from hydro1.sci.gsfc.nasa.gov + 243 positive examples indicated above + 7000 negative examples indicated above)

Used Validation set:

6000 examples

3000 positive examples collected from <ftp://hydro1.sci.gsfc.nasa.gov/data>

3000 negative examples collected from <http://digitalcorpora.org/corpora/govdocs>

Used Test set:

5891 examples:

3061 positive examples collected from <ftp://hydro1.sci.gsfc.nasa.gov/data>

2830 negative examples collected from <http://digitalcorpora.org/corpora/govdocs>

Learning curve:

The following learning curve roughly gives us a feel with how many training example we might need to be able to classify well against the validation set.

The **blue curve** is the learning curve for *validation set*.

The **black curve** is the learning curve for *training set* starting from the 3000th example onwards with the squared error collected every 200 examples each time.

Since the linear logistic regression model seems to do a good prediction in the grb examples based on my last week research findings, this implies that the problem is a linear problem, and the neural network can also start with a very simple structure, it seems that there is no need to use a very computational expensive network structure to fit the data which seems to linear separable.

The following is the network structure.

256 inputs unites

2 hidden unites

1 output unit – binary decision telling if the input histogram is a GRB type or not.

Regularization term lambda: 1



From the above learning curve, we can see that blue curve and black curve intersects around 20 x 200 = 4000 examples; so this roughly implies that the 4000 number of my training examples would be enough for a model that can predict well my validation data. Even though we might a little steady increase after 8000 examples, that still is not a big problem as the difference is not very huge, also notice the validation set and training set have little difference, and the majority of GRB data I have collected all agree at a similar pattern, and this makes it linear separable; By linear separable, it is meant we do not need to have a very complex network structure to solve this problem.

The following shows the performance testing in my test data with 8243 examples.

(I am going to use the percentage of correctly classified as the measure of success)

8243 examples:

"Accuracy: 99.371923" = (number of true positive + number of true negative) / total

"Accuracy: 99.400000" validation set.

5000 examples:

"Accuracy: 99.320998" testing set

“Accuracy: 99.316667" validation set

4000 examples:

"Accuracy: 98.981497" testing set

"Accuracy: 98.950000" validation set

Importantly, it would be interesting to minimize the size of the positive training examples collected from FTP://HYDRO1.SCI.GSFC.NASA.GOV/DATA and see how many positive training examples from the there would be needed to form a good model that predict well in the FTP://HYDRO1.SCI.GSFC.NASA.GOV/DATA. This is important, as we don’t need to collect lots of positive training examples from there.

Training set

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Training error | Validation error | Testing error | Training Accuracy % | Validation accuracy % | Testing accuracy % |
| [3000 -11243] | 0.073992 | 0.119609 | 0.126143 | 99.417758 | 99.400000 | 99.371923 |
| [3100 -11243] | 0.073815 | 0.123760 | 0.130394 | 99.422888 | 99.416667 | 99.439823 |
| [3200 -11243] | 0.073424 | 0.128838 | 0.135633 | 99.453008 | 99.450000 | 99.439823 |
| [3300 -11243] | 0.072872 | 0.135710 | 0.142685 | 99.471299 | 99.483333 | 99.473774 |
| [3400 -11243] | 0.072172 | 0.145082 | 0.152343 | 99.515553 | 99.516667 | 99.490749 |
| [3500 -11243] | 0.071128 | 0.157866 | 0.165619 | 99.548037 | 99.548037 | 99.473774 |
| [3600 -11243] | 0.069600 | 0.177924 | 0.186487 | 99.607535 | 99.583333 | 99.490749 |
| [3700 -11243] | 0.067528 | 0.209692 | 0.219396 | 99.655355 | 99.633333 | 99.507724 |
| [3800 -11243] | 0.064357 | 0.269981 | 0.281923 | 99.664159 | 99.650000 | 99.558649 |
| [3900 -11243] | 0.058765 | 0.421492 | 0.439146 | 98.474946 | 55.950000 | 54.218299 |

From the above table, it is roughly estimated that [3500 -11243] (where 4000-3500 = 500 training examples from the FTP://HYDRO1.SCI.GSFC.NASA.GOV/DATA ) would be enough to well classify the training data.

It is worth noticing that it is ok to use the full data set for training, but there is no such a need, for example, if we have a subject we want to study, a good text book for that subject would be enough, there is no need to study that subject with many similar text book.

Based on the above numbers and table, it is observed that roughly around 500 number of positive training examples would be fairly enough for training, but i would probably tend to include as many negative examples as possible, as negative class is huge and enormous . Although The learning curve roughly shows the validation error is low with 4000 training examples (1000 positive, 3000 negative), the point is that the validation set still cannot be used to represent the whole unseen data, it might be a good idea to include as many different types of file as possible in our negative class.

Notice, I am intentionally simplifying the network structure with 3 layers + 2 hidden units because of the excessive computational space and time required by network training.

We can surely climb up with more hidden units and more layers, but that would cause slow training. The following roughly shows a couple of test results with lambda set to zero. Lambda (i.e. the regularization term used to restrain the syntactical weights from converging too quickly and it can also seen as a ‘penalty’ to the data training, as sometime we don’t want our model to completely fit the training data but hope it can generalize well) closer to zero might lead to overfitting, and larger lambda will cause under-fitting as larger lambda will change the model towards a linear model. So with lambda set to zero, we may lose the generalization a bit but it can correctly classify the training data with almost 100%, and since the training data and validation set data varies a little, so training with zero lambda should be fine.

The following shows the test result with lambda set to zero.

[1] "The training error cost: 0.007720"

[1] "The validation error cost: 0.008192"

[1] "The testing error cost: 0.015682"

[1] "Training Accuracy: 99.832128"

[1] "Validation Accuracy: 99.933333"

[1] "Testing Accuracy: 99.745374"

Interestingly, if our model completely fits our training data, it also fit our validation and testing data, this implies that the patterns in the data collected from the FTP does vary a little or in other words it is highly possible that there is small number of patterns to be classified in there.

We can also boost the fitting by increasing the number of hidden units and number of layers, but it turns out that training with complex networks takes very long time and memory spaces.

The following shows the structure with 10 hidden units and lambda set to zero as well.

[1] "The training error cost: 0.007134"

[1] "The validation error cost: 0.008918"

[1] "The testing error cost: 0.014105"

[1] "Training Accuracy: 99.896694"

[1] "Validation Accuracy: 99.900000"

[1] "Testing Accuracy: 99.813274"

Note if 10 hidden unites is used, we would have 257 \* 10 + 11 number of units, this is huge cost, generally a simple model is preferred if it is able to fit the data well.

There is a small boost but still we can observe some tiny improvement, again this concludes the model fits the data well with complexity of the structure and byte histogram data distribution varies a little too collected from that ftp site.

**Predicting with GRB data collected from other sites than the** [**ftp://hydro1.sci.gsfc.nasa.gov/data**](ftp://hydro1.sci.gsfc.nasa.gov/data)

Problem: GRB data collected from this particular FTP site all have a very similar histogram pattern which make the classification problem a bit easier, (notice that the linear logistic regression model also generalizes well only within the data collected there)

The following experiment is based on the data collected from other site.

I have collected 80 GRB files (859 MB and roughly 20 MB each) from the <ftp://nomads.ncdc.noaa.gov/CFSR/HP_monthly_means/199205/>

Let’s use the following model to test the newly collected 80 GRB data.

Hidden units: 2, lambda = 0

[1] "The training error cost: 0.007720"

[1] "The validation error cost: 0.008192"

[1] "The testing error cost: 0.015682"

[1] "Training Accuracy: 99.832128"

[1] "Validation Accuracy: 99.933333"

[1] "Testing Accuracy: 99.745374"

80 GRB test set result:

note in this test, we only have positive examples, and perdition is made only on those 80 GRB data, and the following shows the result.

[1] "Testing1 Accuracy: 88.750000" # this is the percentage of correctly classified / total

i.e. (71 / 80 = 88.75%) i.e. the true positive = 71, there is no negative example.

Squared error cost: 0.3134035

At the first glance, the result does not look completely unpromising, but notice we only have 80 GRB examples, and the accuracy drops from 99% to 88.75% and this looks like an indication that we are suffering an under-fitting because our training data and validation data does not seem to be very representative for the whole population of GRB data.