CS 5850 Spring 2016

Project Write-up

This project was an extension of the work done in CS5850, Fall 2015. The intent was to develop a deeper knowledge of deep learning neural networks (DLNN) through the use of texts, Python, and one or more deep learning frameworks, such as Theano and or Keras.

bks
5/6/2016

Table of Contents

Beginning: Setup and Texts

Over Winter break we installed the Keras and Theano neural network frameworks on desktop system running Ubuntu 14.04 and two Geforce GT 730 1GB GPUs.

We made use of several texts by Jeff Heaton, primarily relying on his text Introduction to the Math of Neural Networks [1], and Artificial Intelligence for Humans v3 [2], as a structured introduction to the subject of neural networks and deep learning. The texts were a good introduction to the subject and the author's website [3] has material, such as interactive JavaScript pages, to supplement his texts.

Other sources, such as Wikipedia.org, the text Deep Learning [4], and various internet searches were used also.

Middle: Apply CNN to Forest Fire data N-encoding

Near the middle of March we became involved with looking at extending the work of Storer and Green [4], by making use of Convolutional Neural Networks (CNN) on the Montesinho forest fire data set.

The Montesinho forest fire data set had 12 input variables and Storer and Green n-encoded the month and week variables to expand the variables to 29. They also applied the Particle Swarm Optimization algorithm to training the network weights, for a reduction in the root mean squared error (RMSE) as reported in [4].

Two CNN's were used as a basis to build upon and various permutations of a feedforward multilayer perceptrons (MLP) were used in the next section.

End: Models

MLP

Two MLP's were developed and details can be found on GitLab [6]. Each MLP had two versions, one with a fewer number of nodes per layer and the other with a larger number of nodes per layer, as shown in table 1 below.

Table 1 MLP Details

Layers	Nodes	Nodes		
		Layer 1	Layer 2	Layer 3
2	Few	29	15	0
2	Few	25	15	0
2	Few	20	15	0
3	Few	29	15	12
3	Few	25	15	12
3	Few	20	15	12
2	Many	200	100	0
2	Many	100	50	0
2	Many	50	25	0
3	Many	200	100	12
3	Many	100	50	12
3	Many	50	25	12

A comparison of the mean of normalized weather data for the MLP's with few nodes is provided in Table 2 below.

Table 2 Comparison of MLP's with few nodes per layer

Labels	Mean Normed	Layers	Nodes	Epocs	Batch Size	Folds	Activation	
							Hidden	Output
2	0.57	2	Few	300	32	5	relu	softmax
2	0.52	3	Few	300	32	5	relu	softmax
3	0.57	2	Few	300	32	5	relu	softmax
3	0.59	3	Few	300	32	5	relu	softmax
4	0.47	2	Few	300	32	5	relu	softmax
4	0.50	3	Few	300	32	5	relu	softmax
5	0.50	2	Few	300	32	5	relu	softmax
5	0.50	3	Few	300	32	5	relu	softmax
6	0.45	2	Few	300	32	5	relu	softmax
6	0.48	3	Few	300	32	5	relu	softmax
Overall	0.51							

The overall mean percent correct is just over 50%, but both layers had almost 60% accuracy when the data set was partitioned into three labels. Table 3 provides similar information for models with a greater number of nodes per hidden layer of the same models.

Table 3 Comparison of MLP's with many nodes per layer

Labels	Mean Normed	Layers	Nodes	Epocs	Batch Size	Folds	Activation	
							Hidden	Output
2	0.57	2	Many	300	32	5	relu	softmax
2	0.54	3	Many	300	32	5	relu	softmax
3	0.59	2	Many	300	32	5	relu	softmax
3	0.60	3	Many	300	32	5	relu	softmax
4	0.54	2	Many	300	32	5	relu	softmax
4	0.57	3	Many	300	32	5	relu	softmax
5	0.50	2	Many	300	32	5	relu	softmax
5	0.48	3	Many	300	32	5	relu	softmax
6	0.48	2	Many	300	32	5	relu	softmax
6	0.49	3	Many	300	32	5	relu	softmax
Overall	0.54							

The overall average accuracy is greater for this group, with the data sets with three labels doing the best overall. Several attempts were made with both the sigmoid and hyperbolic tangent, but the results were essentially the same, so they've been omitted, but are available in output.xlsx on GitLab/exploringKeras.

CNN

There were five CNN models attempted, but only three were functional before the end of the semester. Table 4 below provides an overview of these models.

Table 4 CNN models attempted

Model	Version	Layers	Activation	
			Hidden	Output
Keras Example	1	8	relu	softmax
(517x1x1x29)	2	Not functional	relu	softmax
Heaton pg. 205	3	9	relu	softmax
Heaton pg. 205 extended	4	11	relu	softmax
Haykin pg. 246 [7]	5	Not functional	relu	softmax

The first model is based on the Keras CNN example [8]. The second was an attempt to move away from reshaping the data from that of an image to that of a vector, but we were unsuccessful at getting it to work properly. The next two examples were taken from the Heaton text [2], and the third was an unsuccessful attempt to mimic an example from Haykin's text [7].

The following tables give summary results for the different models for varying data set labels. As was present in the MLP models, data sets with three labels often showed the best accuracy.

Table 5 Keras Example

Model	Labels	Normalized Accuracy	Layers	Epochs	Batch Size	Folds	Activation	
							Hidden	Output
Keras example	6	0.47	8	300	32	5	relu	softmax
Keras Example	5	0.52	8	300	32	5	relu	softmax
Keras Example	4	0.53	8	300	32	5	relu	softmax
Keras Example	3	0.66	8	300	32	5	relu	softmax
Keras Example	2	0.56	8	300	32	5	relu	softmax

Table 6 Heaton Example

Model	Labels	Normalized Accuracy	Layers	Epochs	Batch Size	Folds	Activation	
							Hidden	Output
Heaton pg. 205	5	0.53	9	300	32	5	relu	softmax
Heaton pg. 205	4	0.54	9	300	32	5	relu	softmax
Heaton pg. 205	3	0.59	9	300	32	5	relu	softmax
Heaton pg. 205	2	0.52	9	300	32	5	relu	softmax
	Overall	0.55						

Table 7 Heaton Example Extended

Model	Labels	Normalized Accuracy	Layers	Epochs	Batch Size	Folds	Activation	
							Hidden	Output
Heaton pg. 205 extended	5	0.50	11	300	32	5	relu	softmax
Heaton pg. 205 extended	4	0.52	11	300	32	5	relu	softmax
Heaton pg. 205 extended	3	0.50	11	300	32	5	relu	softmax
Heaton pg. 205 extended	2	0.56	11	300	32	5	relu	softmax
	Overall	0.52						

Future Work

Although a lot was learned this semester, we ran out of time at the end before running all of the models we desired.

Even though the course is finished, we plan to continue working on this project over the next month or summer if necessary. We would like to give it a through analysis in hopes of accruing benefits in the areas that follow.

Keras

We feel we were just on the cusp of developing some comfort and skill using Keras, and would like to build on this before bring the project to an end.

Neural Networks (General)

We have found Keras to be easy to learn and use, even if the neural network concepts at times are not, and would like to put it to work building various networks in the future. We found Keras's simplicity to be helpful in learning more about neural networks, because well, it just seems to work, so long as you don't build an invalid network. You spend more time working with your models and less time fighting the software, which means more time learning.

As an example, unlike the CNN's, building the MLP's was easy, and the great thing about it was you could quickly start running iterations of you network. We felt like we were starting to get those "a ha" moments when there was less time between having an idea and getting it to run on your framework.

Neural Networks (CNN)

The CNN's in this project can probably be improved, perhaps significantly, as we were unable to deviate from the above examples without quickly running into exceptions and errors. We could mostly add layers and change activation functions, but other changes were unsuccessful. We believe with some time and tinkering, the CNN models could be rebuild with say a 517x1x2x29 array, with one row of zeros instead of 28. We suspect that at the least this would allow for faster convergence and allowing more model iterations to be run.

Currently, the average CNN model was taking about 1 minute each to run. This is tedious when not automating to run a large number of permutations, as was done in the Green / Storer article. Having some comfort with the models we've been using we'd refactor the code into classes of models that could be more easily iterated over with many parameter settings, and outputting information to flat files, or better yet directly to a database for additional analysis.

Go

We would like to read the articles on the training of networks to play the game Go. Intuitively, we believe that the tasks of predicting moves in a board game, and that of recognizing patterns in the forest fire data, have some similarity, some link. Hopefully, the approaches used in these articles will give clues as to changes for the current CNN's on the forest fire data.

Conclusion

We wish to thank Dr. Green for agreeing to advise this project. We hope that with additional work on this project over the summer the CNN models can be improved upon, and have something interesting to report on and hopefully write about in the future. We hope to be able to work on additional projects with Dr. Green in the future also.

References

1. Heaton, Jeff, Introduction to the Math of Neural Networks, 9781604390339
2. Heaton, Jeff, Artificial Intelligence for Humans, Vol 3: Neural Networks and Deep Learning, 1505714346
3. http://www.heatonresearch.com/aifh/vol3/
4. Article provided by Dr. Green, PSO Trained Neural Networks for Predicting Forest Fire Size: A Comparison of Implementation and Performance
5. Deep Learning, unpublished, Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Book in preparation for MIT Press}, http://www.deeplearningbook.org
6. https://gitlab.com/neural-networks/exploringKeras
7. Neural Networks A Comprehensive Foundation, Simon Haykin, 0132733501
8. https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py