Machine Learning and having it Deep and Structured Homework I Report

Group "SYChienIsGod"

April 7, 2015

1 The Group

1.1 Members

The members of the group "SYChienIsGod" are

1. 洪培恒, r02943122
2. 曾泓諭, r03943005
3. 柯揚, d03921019
4. 李啓為, r03922054

1.2 Contribution

As all members were included in the code and algorithm review process (GitHub was used to maintain a common code base), the following list includes only the main efforts of each member:

1. 洪培恒, DropOut, ensemble training, parameter tuning
2. 曾泓諭: stochastic gradient descent/momentum, model ensemble,
3. 柯揚: data preprocessing, basic network structure, report
4. 李啓為, DropConnect, parameter tuning

2 The Algorithm

2.1 Data Preprocessing

For this first homework, the data considered were FBANK as well as the MFCC features to keep computation time low (i.e. raw audio data would have required convolutions which are - although parsimonious in their parameters - expensive to evaluate compared to fully connected layers). Preprocessing steps considered where rescaling and shifting as well as PCA to lower the number of dimensions. It has been widely reported in the literature that data should be centred (i.e. have zero mean) and properly scaled. To achieve proper scaling, he distributions in all 108 dimensions of the data were examined. As all dimension showed a spread in the data below one order of magnitude, a simple linear scaling could be considered (otherwise, more sophisticated scalings like $\log(1 + x)$ would have to be applied). Among the examined factors were max scalings (i.e. scale to $[-1, 1]$), the standard deviation σ and the variance σ^2. Experimental results clearly indicated that simple max scaling was inappropriate as outliers would force the majority of the data to be

close to a constant (e.g. 0) which inhibits learning. While both standard deviation and variance allowed learning to progress, the standard deviation offered a better convergence behaviour and was therefore used in all further experiments.
In addition to that, variance concentration by Principal Component Analysis was considered. The goal is to have fewer variables to capture most of the variance in the data through a linear transformation. Preliminary analysis showed that 45 components of the 69 FBANK variables where necessary to retain over 99% of the variance, however, besides a small speed-up resulting from the reduced number of variables the classification results did not improve, which is why further usage of this technique was discarded from experiments.

2.2 Data Structure

For the usage in Python, the data was converted using Python's cPickle to be able to read the data fast. The lookups to map the FBANK and MFCC features to phoneme labels where performed ahead of the experiments and stored in the pickled data in the same order as the FBANK and MFCC features. As we restricted the experiments to use the features instead of raw data, there was no extensive need for data structure tuning as the data fits in the memory of an average scientific computer.

2.3 Algorithm Design

The current literature offers a lot of inspiration about what to add beyond increasing the layering of tanh-activated networks such as

- Dropout [5, 9] and Dropconnect [11] to inhibit coevolution of nodes
- (Parametric) [4] Rectangular linear units as activation functions [8]
- Batch Normalisation [7] to enhance convergence
- Fast Food Layers [12] and Extreme Learning Machines [6] to thin out the parameter space of fully connected layers (see [3] as well)
- Ensemble methods such as bagging [2] to combine multiple predictors with uncorrelated errors
- Gradient Descent enhancements like momentum [10]

2.4 Implementation

To implement the above algorithm, Theano [1] was used for its general applicability and its simplicity in use in connection with Python. The main source of bugs arose from the requirement to "think in symbolic variables" as in Theano, every operation is defined symbolically to allow for optimisations and device specific compilation. The latter allows to evade Python's slow execution speed by compiling the symbolic expressions to (CUDA) C++.

3 The Experiments

Experimental Results (on GTX670)			
Version	Validation Accuracy	Submit Accuracy	Features
zero	0.586875	0.622	MFCC, 2 layer
0.01	0.595	0.626	MFCC, 2layer, ReLU
0.01-1	0.604	X	FBANK, 3 layer, ReLU
0.01-2	0.614 (@2000 epoch)	X	FBANK, 4 layer, ReLU
0.02	0.602 (@275 epoch)	X	FBANK, 2 layer, PReLU
0.02-1	0.622 (@2000 epoch)	X	FBANK, 4 layer, PReLU
0,03	0.628 (@2000 epoch)	0.63684	as 0.02-1, Momentum
0.03b	0.671 (@2000 epoch)	0.65078	as 0.03, 39 Phonemes
0.03c1	0.682 (@1000 epoch)	X	as 0.03b, 4L wider node
0.03c2	0.690 (@279 epoch)	0.65481	as 0.03b, 5L
0.03d	0.692 (@378 epoch)	0.66202	as 0.03c2, L2Regression
0.03d1	0.700 (@511 epoch)	?	as 0.03d, 7L
0.03e	0.692 (@328 epoch)	0.66081	as 0.03e, FBANK+MFCC
0.03e1	0.706 (@551 epoch)	0.62023	as 0.03e, 7L
0.03g	?	?	as 0.03e1, Learning Rate decay
0.04	0.75 (train accuracy)	0.67867	8 model blending, all 5L FBANK+MFCC
0.04a	?	?	major revision, DropOut
0.04b	0.693	X	DropConnect
0.04e	0.783 (@744 epoch)	0.72683	combine 3 frames

Single Model Detail							
No	Seed	Batch	Input	Epoch	Validation	Training	Layer
x	x	256	F+M	500	0.686	0.726	128/256/512/256/128
x	x	256	F+M	500	0.686	0.723	128/256/512/256/128
x	x	256	F+M	500	0.686	0.721	128/256/512/256/128
x	x	1000	F+M	500	0.692	0.715	128/256/512/256/128
x	x	1000	F+M	500	0.692	0.716	128/256/512/256/128
x	x	512	F+M	500	0.694	0.742	128/256/512/256/128
1.	5556	512	F+M	1071	0.695336	0.749	128/256/512/256D/128D
2.	6789	512	F+M	1164	0.693981	0.747	128/256/512/256D/128D
3.	3324	256	F+M	731	0.688221	0.733	128/256/512/256D/128D
4.	3364	512	F+M	523	0.705	0.759	128/128/256/256/256/128D/64D
5.	4653	128	F+M	265	0.706982	0.751	128/128/256/256/256/128D/64D
6.	3142	128	F+M	?	?	0.724	128/128/256/256/256/128D/64D bagging 0.67
7.	3655	128	F+M	185	0.693393	0.730	128/128/256/256/256/128D/64D bagging 0.67
8.	3242	64	F+M	90	0.691304	0.720	128/128/256/256/256/128D/64D bagging 0.67
9.	3255	64	F+M	129	0.692003	0.727	128/128/256/256/256/128D/64D bagging 0.67
10. by HY							
11.	4982	512	F*3	249	0.730799	0.774	256/256/256/128/128/128D/64D
12.	2312	256	F*3	1413	0.759103	0.777	256D/256/256/128/128/128/64
13.	5342	128	F*3	493	0.771038	0.798	256D/256/256/128/128/128/64
14.	1010	128	F*3	494	0.771393	0.798	256D/256/256/128/128/128/64
15.	1009	128	F*3	492	0.772371	0.802	256D/256/256/128/128/128/64
16.	1321	128	F*7	955	0.805114	0.856	512D/256/256/128/128/128/64 LR=0.08
17.	3453	128	F*7	999	0.807407	0.857	512D/256/256/128/128/128/64 LR=0.12
18.							
19.							

Blending Model Detail			
Recipe	Num of Model	Training	Testing(kaggle)
0-1 0-8	8	0.75	0.67867
1+2+3	3	0.771	0.67533
1+2+4 9	8	0.780	0.69895
11 15	5	0.818	0.74427

References

[1] James Bergstra, Olivier Breuleux, Frederic Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math compiler in Python. In *Proc. 9th Python in Science Conference (SCIPY 2010)*, pages 1–7, 2010.

[2] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, August 1996.

[3] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting Parameters in Deep Learning. June 2013.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. February 2015.

[5] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. July 2012.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications, 2006.

[7] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. February 2015.

[8] Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, January 2014.

[10] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[11] Li Wan and Matthew Zeiler. Regularization of neural networks using dropconnect. *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 109–111, 2013.

[12] Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep Fried Convnets. December 2014.