236757 - Deep learning project
Warm-up exercise

Submission date: 20/05/16

1 Exercises

The following are two exercises on convolutional and recurrent networks. For each exercise you should write your own training code and satisfy the mentioned constraints. The performance parameters for each exercise appears below. You can submit you work only after achieving these performance levels.

1.1 Submission instructions

Submission will be in pairs (project partners) and will contain a short (two pages) pdf report containing:

- Model architecture description, training procedure (data augmentation, regularization, optimization details etc).
- Two convergence graphs - for error and loss as a function of time (epochs). Each graph should depict both training and test performance.
- A short summary of your attempts and conclusions.

In addition, you should also supply:

- Code able to reproduce your results - we might test it on different variants on these datasets.
- The trained network (with trained weights) in Torch format.

1.2 Convolutional networks

Write a complete training procedure for a convolutional network on CIFAR10 dataset. Design and train your network so that it will satisfy the 2 following goals:

- Final accuracy on the test-set should be > 85%
- Number of trainable parameters (weights) within the network should be < 50K

1.2.1 Data

Dataset in Torch format can be downloaded using cifar.torch repository.
1.3 Tips
A useful source (with links to relevant papers) is Cifar - Torch blog entry

1.4 Recurrent networks
Write a complete training procedure for a word-level LSTM recurrent network on Penn Treebank dataset. Design and train your network so that it will satisfy the 2 following goals:

- Final word-level \textit{perplexity} on the test-set should be < 120
- Number of trainable parameters (weights) within the network should be < 3M

In addition to the requirements set, use the networks achieving these goals to generate 5 random sentences continuations to (no apostrophes):

“Buy low, sell high is the” ...

1.4.1 Recurrent implementation
You may use any Torch recurrent implementation (or write your own). Some popular choices:
- Element-Research’s rnn
- torch-rnn
- Implementation shown in class

1.4.2 Data
The Penn Treebank data (in text format, preprocessed to word-level) is available here under:
\texttt{\textbackslash data\textbackslash ptb.train.txt}
\texttt{\textbackslash data\textbackslash ptb.valid.txt}
\texttt{\textbackslash data\textbackslash ptb.test.txt}

1.5 Tips
It will be useful to read the following paper Recurrent Neural Network Regularization.