Deep Learning Teaching Kit

Lab 2

---

1 More Backpropagation

1.1 Backpropagation through a DAG of modules

Figure 1: a Directed Acyclic Graph of modules

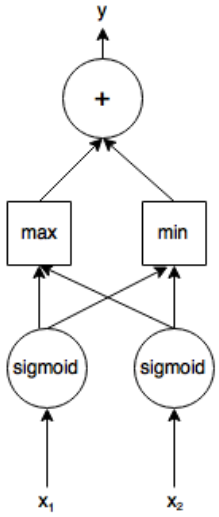

Figure 1 describes a simple network with the inputs being x_1 and x_2 (both scalars). Both inputs are passed through a sigmoid layer. Next, the outputs from the sigmoid layers are passed through max and min layers whose outputs can be represented as $o_{\min} = \min(i_1, i_2)$ and $o_{\max} = \max(i_1, i_2)$. The values output by the min and max layers are then summed in the final layer. Suppose the error passed down (backprop) to y is $\frac{\partial E}{\partial y}$. Write the values for $\frac{\partial E}{\partial x_1}$ and $\frac{\partial E}{\partial x_2}$ in terms of $\frac{\partial E}{\partial y}$.

1.2 Batch Normalization

We'll be working with Batch Normalization (BN) [1] trick for STL-10, which prevents covariate shift across the layers and effectively speeds up the training process. Assume we are about to implement a module that does this trick. The forward pass can be written as:

$$y_k = \frac{x_k - E(x_k)}{\sqrt{(\sigma(x_k))}},$$

where x_k and y_k denote the input batch and output batch, $E(x_k)$ and $\sigma(x_k)$ are mean and standard deviation respectively. Questions:

(i) Given $\frac{\partial E}{\partial y_k}$, write down $\frac{\partial E}{\partial x_k}$, where E is the energy function.

(ii) The vanilla version of BN would normalize the input batch to have mean of 0 and a standard deviation of 1. It might be too restrictive sometimes however. Can you derive a way to incorporate a learnable shift variable into the current formula? Let's assume the considered variable is denoted by ϵ. Please write down the forward and backward pass, and derive $\frac{\partial E}{\partial \epsilon}$.

2 STL-10: Semi-supervised Image Recognition

2.1 Sample code

We provide the `Lab2 sample_code` at `https://bitbucket.org/junbo_jake_zhao/deeplearningkit/src/master/Lab2/sample_code/`. The `provider.lua` file will download the STL-10 dataset from AWS and pre-process the dataset. Please note that more aggressive data augmentation might be useful for STL-10 which has only a small amount of labeled data.

Note that this lab is recommended to be run on a NVIDIA GPUs because CPUs would take a longer time. In Torch, the conversion between CPU and GPU is easy:

```
-- Shift a model from host to device
model:cuda()
-- Shift the criterion from host to device
criterion:cuda()
-- Shift tensor from host to device
data = data:cuda()
-- Note, you need to assign the resulted tensor to a new tensor.
-- However this wouldn't be needed when shifting model or criterion.
```

2.2 STL-10

Quite different from MNIST digit recognition, STL-10 (`https://cs.stanford.edu/~acoates/stl10/`) features a semi-supervised natural image recognition problem. In this lab, you will be using only 5000 labeled images (which is split into 4000 for training and 1000 for validation), and another 100,000 images without labels. All of them are high resolution (96 x 96) RGB images.

You will want to look into some unsupervised/semi-supervised learning approaches to make use of unlabeled data.

2.3 Visualization

2.3.1 Visualizing filters and augmentations

Visualization gives a good perspective on model convergence. One can check if the model is training well or not by taking a look at the filters across the layers. We usually expect the first layer being "edges" or "color blobs". Obviously, these patterns in the filters are noticeable only after the training process is complete. That being said, please include the following pictures in your writeup:

(i) n images visualizing the first layer filters, where n is the number of filters in your first layer, nicely organized in a grid. A good example of such images can be found in section 3 of [2].

(ii) A sample of the augmentations you performed on the dataset, also organized in a grid.

Hint: you will want to look into the "image" module documentation.

2.3.2 t-SNE

Using t-SNE, we can visualize how well our features cluster the images into their respective buckets/ classes. A tutorial on how to use t-SNE in torch for MNIST can be found in the `Lab2` folder in the Bitbucket repository: `https://bitbucket.org/junbo_jake_zhao/deeplearningkit/src/master/Lab2/`. To run this tutorial, download `t-SNE.ipynb`, run `ipython notebook` from the command line, find the file in the interactive interface and then open it.

Your task in this part of the assignment is to use features from your trained model to cluster randomly selected 1000 labeled test images of STL-10. You should include in your writeup an image similar to the digit map at the end of the tutorial. It is a good idea to experiment with features at different layers to discover which layer clusters the best.

2.4 Evaluation

The following metrics will be used to evaluate your model:

- 20% - Kaggle performance. Full score as long as you beat the benchmark.
- 20% - Show improvement by using unlabeled data versus just plain supervised methods; i.e., simply trying to improve the performance of a Convnet on labeled data won't get the score for this part.
- 40% - Writeup on your model structure, required visualization, training process, experiments, results, etc. We expect a rather formal report written with LaTeX.
- 20% - Simple, readable, commented code of final submitted script `result.lua` that is able to execute on the test data and generate a prediction file consistent with your final Kaggle submission.

2.5 Kaggle

You need to save the predictions and submit them to the Kaggle competition. Once you have understood the starter code, try to improve the model and beat the competition. After you finalize your model, create a script `result.lua`, that generates a file named `predictions.csv`. We will check that `predictions.csv` matches your final Kaggle submission. You will need to save your trained model as well to be able to generate the predictions via `result.lua`. The function `torch.save()` in the starter code does exactly this.

DO NOT submit your trained model file (the binary generated by torch during training) via email. These files are generally huge.

DO NOT train on the test data. Training on the test data is considered plagiarism. Please use a validation set for all experiments.

Submission

Send your submission (writeup and `result.lua`) to your corresponding TA by the deadline. Merge your solutions to section 1 with the writeup from section 2. Include a link to the trained model file in the email. Please use the following title for your email.

`[`*CourseName* `YOUR_TEAM_NAME] Submission Lab2`

References

[1] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks*, CoRR, abs/1312.6229, 2013

[2] S. Ioffe, and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, CoRR, abs/1502.03167, 2015.