DS-GA 1008 - Deep Learning, Spring 2017

Assignment 1

Due: Tuesday, March 7th, 2017 at 8:35pm

1 Backprop

1. **Nonlinear Activation Functions:** Sigmoid, tanh, and rectified linear unit (ReLU) are commonly used activation functions in deep learning. They are defined as:

   `sigmoid`:

   $$x_{\text{out}} = \sigma(x_{\text{in}}) = \frac{1}{1 + \exp(-x_{\text{in}})}, \tag{1}$$

   `tanh`:

   $$x_{\text{out}} = \tanh(x_{\text{in}}) = \frac{\exp(2x_{\text{in}}) - 1}{\exp(2x_{\text{in}}) + 1}, \tag{2}$$

   `ReLU`:

   $$x_{\text{out}} = \text{rect}(x_{\text{in}}) = \max(0, x_{\text{in}}), \tag{3}$$

   where x_{in} is the input scalar and x_{out} is the output scalar. Assume the error backpropagated to x_{out} is $\frac{\partial E}{\partial x_{\text{out}}}$. For each activation function, write the expression for $\frac{\partial E}{\partial x_{\text{in}}}$ in terms of $\frac{\partial E}{\partial x_{\text{out}}}$.

2. **Softmax:** Multinomial logistic regression is a generalization of logistic regression into multiple classes. The softmax expression is at the crux of this technique. After receiving n unconstrained values, the softmax expression normalizes these values to n values that all sum to 1. This can then be perceived as probabilities attributed to the various classes by a classifier. Your task here is to backpropagate error through this module. The softmax expression which indicates the probability of the i-th class is as follows:

   $$P(y = i | X_{\text{in}}) = (X_{\text{out}})_i = \frac{e^{-\beta (X_{\text{in}})_i}}{\sum_k e^{-\beta (X_{\text{in}})_k}} \tag{4}$$

   What is the expression for $\frac{\partial (X_{\text{out}})_i}{\partial (X_{\text{in}})_j}$? (Hint: Answer differs when $i = j$ and $i \neq j$).

   The variables X_{in} and X_{out} aren't scalars but vectors. While X_{in} represents the n values input to the system, X_{out} represents the n probabilities output from the system. Therefore, the expression $(X_{\text{out}})_i$ represents the i-th element of X_{out}.

2 Techniques

2.1 Optimization

Please go through section 2 of [1]. Assume that we are minimizing the objective function $f(\theta)$ by gradient descent method. Please write down the mathematical formula of gradient descent step by momentum method and Nesterov's Accelerated Gradient method. And briefly explain the difference between these two techniques.

2.2 Reducing Overfitting

1. **Dropout** [2] one of the most effective technique to regularize a fixed-sized model. The key idea is to randomly drop units (along with their connections) from the neural network during training. **Ensembling** is a general term for combining many classifiers by averaging or voting. Please explain how Dropout model can be visualized as Ensembling model?

2. Consider any simple neural network (see section 4 of [2] to get the idea of sample network) of your choice with dropout probability (keep probability) p and briefly explain your model behavior in terms of hidden layer weight scaling during Training and Testing time?

3. Deep Neural nets for image classification generally require a large and rich image data set to train well without overfitting. Many times you would come across with the Data Scarcity problem which can lead your deep model to overfit. **Data Augmentation** [1] is one of the most common technique to solve this problem. Which data augmentation methods would you apply for MNIST dataset and why? Briefly explain the methods.

2.3 Initialization

When you try to minimize a non-convex function using stochastic gradient descent (SGD), the solution depends on the initial values of the parameters. For convolutional neural networks (CNNs), the widely used method is to initialize weights using zero-mean gaussian random variables with standard deviation 0.01 [3]. Biases could be initialized to 0, 1, or your choice. Some previous work introduces better ways to initialize parameters. Read the cited papers and answer the following questions.

1. Briefly explain the initialization method introduced in He et al. (2015)[4]. How is it different from Xavier initialization [5]?

2. Pretraining is another way to initialize your neural network. How did the VGG team use pretraining in there model [6]? How did Coate et al. (2011) [7] extract features from unlabeled data? Develop and explain your approach for the semi-supervised version of MNIST (Part 3).

[1] http://neuralnetworksanddeeplearning.com/chap3.html (see section "artificial expansion of the training data")

3 PyTorch (MNIST Handwritten Digit Recognition)

Semi-Supervised Learning: You will be on a journey to explore the MNIST digit recognition task [2] in this assignment, while a large portion of the labels are lost. We restricted the size of labeled data samples to 3,000. Meanwhile, 57,000 digit images are provided without the corresponding label. This problem setting is called "semi-supervised learning". One needs to find a way to leverage the unlabeled data, on a basis of a discriminative neural network trained on the labeled data in the classical way, to obtain best classifier performance. We recommend the following papers:

1. *Ladder Network:
   `https://arxiv.org/abs/1507.02672`
2. *VAE:
   `https://arxiv.org/abs/1406.5298`
3. SWWAE:
   `http://arxiv.org/pdf/1506.02351v8.pdf`
4. Universum:
   `http://arxiv.org/pdf/1511.03719v5.pdf`
5. Pseudo-label:
   `http://deeplearning.net/wp-content/uploads/2013/03/pseudo_label_final.pdf`
6. Surrogate class:
   `http://papers.nips.cc/paper/5548-discriminative-unsupervised-feature-learning-with-convolutional-neural-networks.pdf`
7. Sparse auto-encoder:
   `https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf`
8. Denoising auto-encoders:
   `http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf`

STL-10 dataset another semi-supervised learning dataset in the image domain, this list would be useful to refer to: `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#53544c2d3130`.

Starter Code: `https://github.com/jakezhaojb/DSGA-1008-Spring2017-A1`. The sample code is based on `https://github.com/pytorch/examples/tree/master/mnist`. The difference is that we are using only 3,000 labeled data to train the model. It is your mission to use the rest unlabeled data to improve the performance.

Kaggle Leaderboard: You need to save the predictions into a `.csv` file and submit them onto Kaggle (now the GAME IS ON!). Once you have understood the starter code, try to improve the model performance and beat the baseline results by a maximal margin. Please

[2] `http://yann.lecun.com/exdb/mnist/`

keep in mind that you are allowed to use **CPU** only. After you finalize your model, create a script `mnist_model.py`, which is used for training your best submitted model. Alongside the training script, one also need to submit a script `mnist_result.py` that dumps out a file named `predictions.csv` using your model on the testing set. We'll verify the replicability. **DO NOT** submit your trained model file via email.

Evaluation: Your grade for this section will be based on:

- 25% - Kaggle performance. Full score as long as you beat the benchmark.
- 15% - Simple, readable, commented code of final submitted script `mnist_model.py` and `mnist_result.py` that is able to execute on the test data and generate a prediction file consistent with your final Kaggle submission.
- 25% - Write up : Section 1 (10%) , Section 2.1 (5%) , Section 2.2 (5%) , Section 2.3 (5%)
- 10% - Explain your final model architecture and configuration of your model
- 10% - How did you improve the accuracy using the techniques from section 2. Share your results with train/validation/test accuracy plots.
- 15% - How did you improve the performance using semi-supervised techniques? Share your results with train/validation/test accuracy plots.

Submission

Send your submission (writeup, `mnist_model.py` and `mnist_result.py`) to `dsga1008.spring.2k17@gmail.com` in a single zip folder named as "YourTeamName.zip" by the deadline. Include a link to the trained model file in the email. Please use the following title for your email.

```
[DS-GA-1008 YOUR_TEAM_NAME] Submission A1
```

We will be sending out Kaggle invitations as soon as you form a team. Please compete as teams on Kaggle with exactly the consistent team names.

References

[1] Sutskever, I., Martens, J., Dahl, G. and Hinton, G: *On the importance of initialization and momentum in deep learning*, ICML (3) 28 (2013): 1139-1147.

[2] Srivastava, Nitish, *Dropout: a simple way to prevent neural networks from overfitting.*, Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, CoRR, http://arxiv.org/abs/1502.01852, 2015.

[5] Xavier Glorot and Yoshua Bengio, *Understanding the difficulty of training deep feedforward neural networks*, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010.

[6] Karen Simonyan and Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, In ICLR, 2015.

[7] Adam Coates, Andrew Y. Ng and Honglak Lee, *An Analysis of Single-Layer Networks in Unsupervised Feature Learning*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011.