CS/CNS/EE/IDS 165 Winter 2020
Instructor: Anima Anandkumar California Institute of Technology
Due 2nd Mar 2020

Homework 3: Graphical Models & Tensor Operations

1 **(Equivalent Markov Properties)** In the lecture, we studied three kinds of Markov property: local, pairwise and global. Show that under the positivity condition, they are equivalent.

Hint: To show that the local-Markov property implies global property use the Hammersley-Clifford theorem to obtain the distribution of a Markov random field and then group the terms.

2 (**Train Convolutional Neural Networks with much less number of parameters.**) Convolutional neural networks (CNN) typically consist of many convolutional layers followed by several fully-connected layers. While convolutional layers map between high-order activation tensors, the fully-connected layers operate on flattened activation vectors. Despite its success, the use of flattening + fully connected layers has notable drawbacks. Flattening discards multilinear structure in the activations, and fully-connected layers require many parameters. We can address these problems by incorporating tensor algebraic operations that preserve multilinear structure at every layer.
Specifically, we do so by replacing the traditional flattening + fully-connected layers with a tensor regression applied directly to the high-order input and enforcing low rank constraints on the weights of the regression [1].
To test the idea of tensor regression layer, we will use Tensorly, a high-level API for tensor methods and deep tensorized neural networks in Python [2]. In this problem, we will train a CNN on **CIFAR10** dataset with TRLs and compare with CNNs trained on the number of parameter needed. You can use Google Colab for this question.

(a) Train a CNN that achieves 74% or higher accuracy. You can use any library you want. An example CNN using Pytorch is given in Resources Section of Piazza. You can modify that code to achieve the sufficient test accuracy. Feel free to use other resources for this question, but please cite the resource or open source code that you use. Here is a tutorial on using pytorch to train a classifier on CIFAR10 which could be helpful.

(b) How many parameters do you need for the fully-connected (FC) layers to achieve this accuracy?

(c) Replace the flattening layer and FC layer with TRLs and train the CNN. An example on MNIST dataset is given here (Note: you can ignore the FIX THIS comment in the code, but there are a few bugs elsewhere in the implementation of tensor regression. Make sure to fix these issues so that training runs correctly). You may want to play around with the ranks in TRL layer and its regularization strength. You should be able to get accuracy higher than 74%. Plot the training and testing loss over epochs.

(d) Report the number of parameters in the CNN and the classification accuracy versus different hyperparameter configurations (ranks, TRL penalty regularization, etc.).

(e) (Bonus) Explore the use of Tensor Contraction Layers (TCLs). While TRLs impose low-rank constraints on the regression weights, TCLs impose low-rank constraints on the activations. You can check [1] for more information. Use Tensorly to implement TCL and report the number of parameter reduction.

References

[1] Jean Kossaifi, Zachary C. Lipton, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. Tensor regression networks, 2017.

[2] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python, 2016.