CSE 250B Project 4B

This project is an alternative to the recursive auto encoder (RAE) project; choose either that project or this one. For either project, you must work in a team of exactly two students, and the due date is Monday March 17, 2014. Please place a printout of your report in Prof. Elkan's mailbox on the second floor of the CSE building before 10am on March 17.

The objective of this project is to develop a method for training multilabel classifiers that leads to better accuracy than the best existing published method. Specifically, the goal is to combine the best ideas from the two recent papers [Kumar et al., 2013] and [Nam et al., 2013]. You may begin with the code that is available at `https://github.com/abhishek-kumar/NNForMLL` for the former paper, or you may do a new implementation in Matlab or in Python. Extend this code to include the following ideas from [Nam et al., 2013]: rectified linear units to make learning faster, the Adagrad method to set stochastic gradient learning rates, and the dropout technique to prevent overfitting. Keep the skip layer network design from [Kumar et al., 2013]. As necessary, use the tips and tricks explained in [LeCun et al., 1998].

Keep your computational experiments as simple as possible. Pick one dataset each from the papers [Kumar et al., 2013] and [Nam et al., 2013]. Choose datasets on which the methods from those papers do especially well. Make sure that you are using the exact same datasets and experimental protocols as in previous papers. In particular, verify that you have an identical number of features and that the cardinality of training, validation, and test sets is identical. Pay attention to details in the previous papers about data preprocessing. If anything is not reproducible about previous experiments, explain the issue clearly in your report. If you can, replicate experiments in previous papers. However, it is more important to implement and refine the new, improved method.

Use the log loss objective function for training, in order to make predictions be well-calibrated conditional probabilities. In experiments, focus on maximizing macro F1 and micro F1 more than on optimizing other measures of performance. Do not use the regression method of setting thresholds. Instead, sort predicted probabilities in decreasing order and use the method of choosing a threshold described in class.

Make sure that your software is fast on a single core. Identify the inner loop of your code and measure its speed approximately in gigaflops. This should be a significant fraction of the theoretical peak performance of your processor. If you use Matlab, the inner loop must be a matrix-vector operation. If you use Python, the inner loop must be compiled, not interpreted.

Before you do experiments with large datasets, make sure that your code is fully correct using small datasets. Do not put yourself in a situation where your work is delayed by needing to wait for computations to terminate.

In the report, give the equations for the partial derivatives that are needed for training. Make sure that backpropagation computes these derivatives correctly. In particular, compare backpropagated derivatives with the result of numerical differentiation of the loss function. The relative difference should be less than 10^{-6} between each numerical derivative and the corresponding backpropagated derivative. To reduce error, use central differences for numerical derivatives.

In the report, analyze the time complexity of computing derivatives numerically and with backpropagation. How can you check that numerical and backpropagation derivatives are equal efficiently? Also explain in the report other steps that you take to verify that your code is correct.

One question whose answer is not clear from the paper [Nam et al., 2013] is whether or not regularization and/or early stopping are useful in addition to the dropout technique. Try to answer this question experimentally in a clear and definitive way. Relevant published papers may exist; look for these.

References

[Kumar et al., 2013] Kumar, A., Menon, A. K., and Elkan, C. (2013). Neural network models for multilabel learning. Accepted for publication subject to revisions. Available at `http://cseweb.ucsd.edu/~elkan/250B/NeuralNetsForMLL.pdf`.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Orr, G. B., and Muller, K.-R. (1998). Efficient backprop. In Orr, G. B. and Muller, K.-R., editors, *Neural Networks: Tricks of the Trade*. Springer Verlag.

[Nam et al., 2013] Nam, J., Kim, J., Gurevych, I., and Fürnkranz, J. (2013). Large-scale Multi-label Text Classification–Revisiting Neural Networks. *ArXiv e-print 1312.5419*.