Machine Learning 6.867 - Project Proposal
Improving Neural Networks via Robustifcation, Dropout, and other Approaches

Ying (Daisy) Zhuo

November 11, 2015

For many challenging machine learning tasks such as image recognition and natural language processing, artificial neural networks have been shown to have very strong performance. This is largely due to its ability to approximate non-linear functions arbitrarily well and to extract patterns from complicated or noisy data. However, the generalization performance of neural networks can be severely impacted by overfitting the training set. In this project, I would like to explore some opportunities to improve neural networks.

0.1 Robust Neural Networks

Regularization is classical approach to reduce overfitting in many machine learning methods. By adding a penalty on the weight parameters to the objective function, the weight parameters are shrunk toward zero. Smaller weights give smoother network response and can potentially reduce overfit. One simple regularizer is *weight decay*, which adds a $\lambda \mathbf{w}^T \mathbf{w}$ term to the error function. In a probabilistic view, simple weight decay regularization is equivalent to putting a Gaussian prior over the weight parameters.

Unfortunately, simple weight decay does not achieve the desirable property of *consistency*, also known as scaling invariance. The invariance property says if we linearly transform the input or target vector, we should obtain equivalent network with linear transformation of weights as given. However, the simple weight decay regularizer term does not maintain this property, as it penalizes on all weights equally. One scale-invariant regularizer is to use different trade-off parameters for the two weights; i.e., adding $\lambda_1 \|\mathbf{w}^{(1)}\|^2 + \lambda_2 \|\mathbf{w}^{(2)}\|^2$ to the negative log likelihood, biases excluded. In the Bayesian framework, putting different priors on $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ corresponds to an improper prior, which cannot be normalized. Using improper priors can lead to difficulty in selecting regularization coefficients and Bayesian model comparisons.

Given the somewhat ad hoc nature of adding regularizers and difficulty in the Bayesian framework on neural networks, I would like to consider the issue from a different perspective. In a recent paper, Bertsimas et al. proposed a principled approach using Robust Optimization to model data uncertainty, and have shown its equivalence to regularization in certain circumstances [1]. For example, by modeling uncertainty in input features on hinge-loss minimization, it is equivalent to support vector machines. The computational results in the paper suggest that systematically addressing uncertainties generalizes better than regularization, as demonstrated in logistic regression, support vector machine, and decision trees.

In this project, I would like to explore the feasibility of robustifying neural networks. My hypothesis is that it is possible, and can be shown to resemble regularization under specific choice of uncertainty

sets. Since robust methods inherently consider data being altered, it should not have issue with scaling invariance as simple weight decay does. I will discuss the computational tractability of robust neural networks. Further, I will test it on multiple datasets. My research group has collected 70+ datasets from UCI Machine Learning Repository with classification tasks, and I plan to test the magnitude of improvements on a sizable subset.

0.2 Augmented Training Data, Dropout, etc.

Many other practical approaches to improve neural networks exist. One method is to artificially expand the training data, knowing that the model should be invariant to translation and scaling. This can be computationally expensive, but is easy to implement and can achieve good results. For example, Simard et al. took the original hand-written digits from MNIST, rotated, translated, skewed the images and added to the samples [3]. They achieved an accuracy of 99.3% in the end.

Dropout has also been shown to be a successful method, where a random subset of the hidden nodes are taken when training, and the prediction is averaged over multiple such thinned networks [2, 4]. The authors applied dropout to MNIST digit classification and achieved an impressive 98.4% accuracy, and 98.7% when combing dropout and a form of L_2 regularization.

If time permitting (or if robust neural network doesn't work), I will implement these methods and replicate the improvement in MNIST data. To validate the advantage of these methods, I would also run the method on a few more data sets.

0.3 Timeline

- Nov 12 - Proposal
- Nov 16 - Discussion; Derive analytic results for robust neural networks
- Nov 23 - Prototype of robust neural network
- Dec 1 - Test on multiple datasets
- Dec 8 - Final project report

If robust neural network doesn't work out, the fall back plan is to implement agumented training data and dropout methods on the week of November 23.

References

[1] Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, and Daisy Zhuo. Robust classification. *Submitted for publication*, 2015.

[2] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[3] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE, 2003.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.